On Efficiently Calculating Small Solutions of Systems of Polynomial Equations

Lattice-Based Methods and Applications to Cryptography

Dissertation
zur Erlangung des Doktorgrades
der Naturwissenschaften
der Fakultät für Mathematik
der Ruhr-Universität Bochum

vorgelegt von
Dipl.-Math. Maike Ritzenhofen

Januar 2010

Reviewer:

Prof. Dr. Alexander May (Ruhr-University of Bochum)

Prof. Dr. Hans Ulrich Simon (Ruhr-University of Bochum)

Contents

Chapter 1

Introduction

Cryptology is an old science, its roots reaching back to the ancient Romans and Greeks. It can be divided into two branches, cryptography and cryptanalysis. Whereas the main interest of cryptographers is to design new cryptosystems, cryptanalysts try to break them. The developments of both directions of cryptology are, of course, strongly related. The development of a new cryptosystem offers new problems to attack, and a newly developed attack leads to the need to develop new cryptosystems.

At its inception, cryptography stood mainly for enciphering and deciphering. One assumed a cryptosystem to be secure as long as no attack was known. In the cause of time, requirements concerning cryptosystems emerged. Different types of security models were defined, and one tried to prove the security of a cryptosystem on certain assumptions.

Until 1976, there was only one type of cryptosystem known as symmetric cryptosystems. In a symmetric cryptosystem two communication partners, say Alice and Bob, have a common secret k. We call k the key. Using k and an encryption function E, Alice encrypts a message m as $c := E_k(m)$ and sends c to Bob. Bob obtains the message by using a decryption function D and computing $m = D_k(c)$. For a symmetric encryption scheme to work correctly, it has to verify $m = D_k(E_k(m))$. A third party, Eve, eavesdropping on the communication should find it difficult to obtain the message m or even useful information on m from c without knowing the key k.

A second type of cryptosystem known as asymmetric cryptosystems was introduced by Whitfield Diffie and Martin Hellman in 1976 [DH76]. They proposed a method of communicating privately without having to share a secret key beforehand. That is, they developed a way to exchange keys via an insecure channel.
The development of the Diffie-Hellman key exchange protocol encouraged the development of asymmetric cryptosystems in the following years. The principle of an asymmetric cryptosystem works as follows: In contrast to symmetric systems, only the receiver Bob has to keep a secret sk. We call sk the secret key. Based on the secret key, Bob constructs a public key pk. This key is published in a way that it is guaranteed that this public key belongs to Bob. The key pk may be seen by anybody. When Alice wants to send a message

to Bob, she takes Bob's public key pk, encrypts the message as $c := E_{pk}(m)$ and sends c to Bob. Then Bob decrypts the message as $m = D_{sk}(c)$. Thus, he needs his secret key to recover m. As in symmetric cryptography, for an asymmetric encryption scheme to work correctly, it has to verify $m = D_{sk}(E_{pk}(m))$. The tuple (pk, sk) should form a pair of corresponding public and private keys. Furthermore, for a third party, Eve, eavesdropping on the communication, it should be difficult to obtain the message m from c without knowing the secret key sk.
One of the main asymmetric cryptosystems was developed by Ron Rivest, Adi Shamir and Leonard Adleman in 1977 [RSA78]. It is still one of the most widely used asymmetric cryptosystems today.

It works on the following principle. Bob chooses two large primes p and q of equal bitsize and defines the value $N := pq$. Furthermore, he chooses $e \in \mathbb{Z}_N$ such that e and $\varphi(N) = (p-1)(q-1)$ are coprime. Then he calculates d such that $ed \equiv 1 \pmod{\varphi(N)}$. Subsequently, Bob publishes (N, e) as his public key and keeps (N, d) as his private key. The encryption function is defined as $E_e(m) := m^e \pmod N$ for $m \in \mathbb{Z}_N$. The decryption function is defined as $D_d(c) := c^d \pmod N$. As $ed \equiv 1 \pmod{\varphi(N)}$ the computation verifies $D_d(E_e(m)) \equiv (m^e)^d \equiv m \pmod N$. We denote the variant defined here by **plain RSA**. For security reasons, it is modified for practical use, e. g. by padding the message. The standard of RSA encryption is given in [RSA].

The security of cryptosystems is proven on the assumption that an underlying problem is difficult. In the case of RSA the underlying number theoretic problem is the so called **RSA problem**: Let $N = pq$ be the product of two large primes p and q, and let e be a positive integer such that $\gcd(e, \varphi(N)) = 1$. Then given N, e and an element x, the challenge is to determine the unique element $y \in \mathbb{Z}_N$ such that $y^e \equiv x \pmod N$.

The RSA problem is assumed to be difficult to solve. That is, we assume that there is no efficient algorithm that solves the problem. Indications for this are given by Ivan Damgård and Maciej Koprowski [DK02]. They show that solving the RSA problem with generic group algorithms is intractable in groups of unknown order. However, the model they use is quite restrictive as it only allows for the use of multiplications.

The **problem of factoring** is closely related to the RSA problem and is defined as follows: Let $N = pq$ be the product of two large primes p and q. Given N, the challenge is to determine its factors p and q.

So far, we have defined the problem of factoring only with respect to moduli which are products of two primes. We call these moduli RSA moduli. One can generalize the definition to include arbitrary composite integers.
It is easy to see that the difficulty of the RSA problem implies the difficulty of factoring. That is to say, if we can factor an RSA modulus N, we can use p and q to compute $\varphi(N)$ and then d. Calculating x^d then gives us the required element y. The opposite, however, is not clear. Partial results are known in restricted models. On the one hand, Dan Boneh and Ramarathnam Venkatesan [BV98] as well as Antoine Joux, David Naccache and Emmanuel

Thomé [JNT07] provide evidence against the equivalence of breaking the RSA problem and factoring, whereas Daniel Brown [Bro06], Gregor Leander and Andy Rupp [LR06] as well as Divesh Aggarwal and Ueli Maurer [AM09] give arguments in favor of the equivalence. The models used in any of the works are restrictive as the proofs are given with respect to straight line programs ([BV98, Bro06]) or generic ring algorithms ([LR06, AM09]). In [JNT07] the attacker is given additional sub exponential access to an oracle determining e-th roots of integers of the form $x + c$. Moreover, the works of [BV98, Bro06, LR06] only consider a special version of the RSA problem with low exponents.

There are, thus, two possible ways of attacking the RSA problem. One can either attack the problem directly or attack the factorization problem. Hence, let us have a more detailed look at the difficulty of the problem of factoring. On the one hand, Peter Shor's algorithm from 1994 [Sho94] demonstrates that the factorization problem is polynomial time solvable on quantum Turing machines. On the other hand, it seems to be highly unclear whether these machines can ever be realized in practice. Furthermore, on classical Turing machines only super polynomial algorithms are known, the Quadratic Sieve [Pom84], the Elliptic Curve Method [Len87] and eventually the Number Field Sieve [LHWL93].

A different line of research to analyze the difficulty of factoring is an oracle-based approach. It was first studied at Eurocrypt 1985 by Ron Rivest and Adi Shamir [RS85], who showed that $N = pq$ can be factored given an oracle that provides an attacker with bits of one of the prime factors. The task is to factor in polynomial time by putting as few queries as possible to the oracle. Ron Rivest and Adi Shamir showed that $\frac{2}{3} \log p$ queries suffice in order to factor efficiently.

At Eurocrypt 1992, Ueli Maurer [Mau92] allowed for an oracle that is able to answer any type of questions by YES/NO answers. Using this powerful oracle, he showed that $\epsilon \log p$ oracle queries are sufficient for any $\epsilon > 0$ in order to factor efficiently. This algorithm, however, is probabilistic. At Eurocrypt 1996, Don Coppersmith [Cop96a] in turn improved the Rivest-Shamir oracle complexity to $\frac{1}{2} \log p$ queries. Don Coppersmith used this result to break the Vanstone-Zuccherato ID-based cryptosystem [VZ95] that leaks half of the most significant bits of one prime factor.
In what follows we will further pursue this goal and try to attack RSA by either attacking instances of the problem itself or by attacking instances of the factorization problem. Therefore, we transform a problem into a polynomial equation or a set of polynomial equations. Then we calculate the solutions or conditions on which the solutions can be determined. The main techniques we use to do so are based on lattices. A lattice is a discrete Abelian subgroup of $\mathbb{Z}^k$.
We use a given set of equations to define a lattice which contains the values we are searching for. If these values fulfill certain properties, e. g. correspond to a small basis vector, we can apply methods to analyze a lattice to determine our target values. We concentrate on a special method introduced by Don Coppersmith in 1996 to find small solutions of a single equation [Cop96b, Cop96a] and analyze on which conditions we can generalize his approach to systems of equations. Hence, the main goal of this thesis is twofold. On the

one hand we would like to apply lattice-based methods and attack special instances of problems related to RSA and factoring. On the other hand, we would like to improve the theoretical knowledge of how to solve systems of multivariate equations with lattice-based techniques.
We will now present the organization of the thesis in more detail.

In Chapter 2 we give the mathematical background. It comprises three sections. Section 2.1 deals with the notation and basic linear algebra. In Section 2.2 multivariate polynomial rings are introduced. Furthermore, some algebraic techniques to solve systems of multivariate equations are presented. In Section 2.3 we define lattices and show how they can be used to determine solutions of a multivariate polynomial equation.

Chapter 3 deals with modular univariate systems of equations. Systems of equations with one common modulus have already been analyzed in [CFPR96]. It is shown that in most cases solutions of such systems can be determined efficiently with algebraic methods. Systems with mutually coprime moduli have been analyzed by Johan Håstad in [Hås88]. As a main result we improve on the bound obtained using Johan Håstad's approach combined with the techniques presented by Don Coppersmith. That is, we present a method to solve systems of modular univariate equations with coprime moduli. The result is obtained due to special polynomial modeling. By this, we can determine larger solutions than one could determine with the previously known methods. We obtain the following result. Let $N_1, \ldots, N_k$ be mutually coprime moduli. For $i = 1, \ldots, k$ let $f_i(x) \equiv 0 \pmod{N_i}$ be an equation of degree δ_i. Then we can determine all solutions x_0 such that $|x_0| \leq X$ and $f_i(x_0) \equiv 0 \pmod{N_i}$ for all $i = 1, \ldots, k$ efficiently if $X \leq \prod N_i^{\frac{1}{\delta_i}}$. This implies that any solution can be determined if $\sum_{i=1}^{k} \frac{1}{\delta_i} \geq 1$.
Moreover, we give an argument why the new bound is optimal for general systems by giving an example of a system for which the bound cannot be improved.
A result with respect to our second goal, namely that of attacking specific problems, is that we can apply the general technique to an RSA broadcast scenario. A passive attacker can determine a message m sent to k different users with coprime moduli N_i and public keys (N_i, e_i), $i = 1, \ldots, k$ if $\sum_{i=1}^{k} \frac{1}{e_i} \geq 1$.
The results presented in this chapter are joint work with Alexander May and published in [MR08].

In the following chapters we deal with general multivariate systems of equations.
In Chapter 4 we treat modular systems of equations with a common modulus. Their solutions are determined by solving a shortest vector problem in a suitable lattice. As a main contribution we introduce and analyze the problem of implicit factoring. This problem fits into the framework of oracle-based factoring. For a given composite number $N_0 = p_0 q_0$ with q_0 being an α-bit prime, our target is to compute p_0 and q_0. During the attack we are allowed to query an oracle which on each query answers with another RSA modulus $N_i = p_i q_i$ such that q_i again is an α bit prime and p_i and p_0 share t least significant bits. We show that with k oracle queries we can heuristically factor all the N_i on condition that

$t \geq \frac{k+1}{k}\alpha$. In case of only one oracle query, the method is provable on the slightly stronger constraint that $t > 2(\alpha + 1)$.
This result is joint work with Alexander May and published in [MR09]. To our knowledge, it is the first result which shows that only implicit information is sufficient in order to factor efficiently. This implies that already a weak form of an oracle is in itself sufficient to achieve a polynomial time factorization process. This gives further insight into the complexity of the underlying factorization problem.

Chapter 5 divides into two parts. The contribution of Section 5.1 is of rather theoretical nature. We try to generalize the algorithm of Don Coppersmith to systems of equations. We denote this algorithm by Coppersmith's algorithm. However, in the process of a straightforward generalization of Coppersmith's algorithm problems occur. One basic step in the original algorithm is the construction of a sublattice with the same determinant as the original one. This step does not necessarily work with a general system. Therefore, as a main result, we introduce the new notion of being **determinant preserving** and state a necessary and sufficient condition of this. This contributes to the theoretical understanding of Coppersmith's algorithm.
In the following two sections we apply the theoretical results to modular systems of polynomial equations and give examples of this. Section 5.2 deals with systems of equations with one common modulus. We show how the necessary and sufficient conditions apply to such systems of equations. Subsequently, we use these observations with respect to RSA with related messages and implicit relations as well as with respect to RSA with random paddings.
In RSA with related messages a set of k secret messages $m_1, \ldots, m_k$ is encrypted with an RSA public key (N, e). That is, we get k encryptions $c_i \equiv m_i^e \pmod{N}$. Furthermore, we have some implicit polynomial relation $p(m_1, \ldots, m_k) \equiv 0 \pmod{N}$. In this context, we apply the generalization of Coppersmith's algorithm to equations in independent variables. If one considers only the condition on the size of the variables we get in this way, one could hope to be able to determine more solutions than with a separate analysis of the equations. This, however, is not the case. We prove the equivalence of both approaches. This result is quite plausible. However, we do not know of any proof of this in the literature published in this subject.
Furthermore, we develop a general strategy on how to apply the generalization of Coppersmith's algorithm to specific systems of independent equations with additional equations providing relations between the unknowns. Unfortunately, the construction is not generic.
The problem of RSA with random paddings is defined as follows. For a user's public key (N, e) with N some n-bit number, a message m of $\alpha < n$ bits is encrypted as $c_i \equiv (2^{\alpha+\tau} v_i + 2^\tau m + w_i)^e \pmod{N}$ for a random τ-bit number w_i and a random $(n - \alpha - \tau)$-bit number v_i and $i = 1, \ldots, k$. The value $k \in \mathbb{N}$ denotes the number of different encryptions. The message m as well as the values v_i and w_i are secret and not known to any attacker. Given two encryptions with different paddings, the message can be determined by combining techniques presented in [Jut98, Cop96b, CFPR96].
Usually, additional information obtained from additional encryptions enables us to de-

termine the solutions on weaker conditions. In this section we analyze two strategies to improve the conditions with more equations and argue why these strategies are not successful. This indicates that we cannot always make use of additional information and contradicts what one might intuitively expect. Moreover, by the given arguments, we indicate a way of using the property of being determinant preserving to prove impossibility results.
In Section 5.3 we consider systems of equations with coprime moduli. One approach to determine solutions of these systems of equations is to reduce the problem to the problem of calculating solutions of a single multivariate equation. This is done by applying the Chinese Remainder Theorem to the original system of equations. Then we can analyze the resulting equation.
We compare this approach to directly analyzing the system with a generalized version of Coppersmith's algorithm. Interestingly, in systems of equations with coprime moduli the necessary and sufficient conditions for the sublattice construction to work become significantly simpler than in the general case. We use this observation to prove the equivalence of the generalized Coppersmith's algorithm and the method via the Chinese Remainder Theorem for univariate equations on some rather natural additional assumptions. In the case of multivariate equations, however, this equivalence does not hold. Any solutions determined with the method based on the Chinese Remainder Theorem can also be obtained with a generalization of Coppersmith's algorithm. For the opposite implication, though, we give a counterexample.

Finally, in Chapter 6 we consider systems of equations over the integers. This case is the most complex one as even the necessary and sufficient condition for the sublattice step in the generalization of Coppersmith's algorithm to work cannot be checked for efficiently. We introduce an additional precondition to obtain a more helpful necessary and sufficient condition. Keeping these constraints in mind, we return to the problem of implicit factoring. That is, we intend to factor an integer $N_0 = p_0 q_0$, where N_0 is an n-bit composite number and q_0 an α-bit prime. In contrast to what is done in Chapter 4, we now allow for the shared bits to be at any position in p_0 as long as they are subsequent. Then we analyze the problem over the integers. We show how to use this method to reconstruct the bound $t > \frac{k+1}{k}\alpha$ we have already obtained with this method using k oracle queries. The value t corresponds to the number of shared bits.
Furthermore, with only one oracle query but a large lattice, we can improve the bound to $t > 2\alpha\left(1 - \frac{\alpha}{n}\right) + \frac{3}{2} + \epsilon$ for some $\epsilon > 0$. Using a similar approach with further oracle queries, we can only reproduce the latter bound. Unfortunately, we cannot improve on it. Any potential strategy for this violates the property of being determinant preserving. When adapting the strategy such that the necessary and sufficient condition is fulfilled, the bound we obtain as a final result gets worse. We claim that a reason for these observations is the special structure of the polynomials we consider. This claim is supported by a counting argument. Hence, this chapter gives an introduction to the use of a generalized variant of Coppersmith's algorithm with equations over the integers, but also indicates open problems and directions for future research.

Chapter 2

Mathematical Background

2.1 General Notation

We start by introducing the notation which we will use in the following chapters. Let $\mathbb{N}$ be the set of positive integers, $\mathbb{N}_0 := \mathbb{N} \cup \{0\}$ and $\mathbb{Z}$ be the ring of integers. For $N \in \mathbb{N}$ let $\mathbb{Z}_N$ denote the ring $\mathbb{Z}/N\mathbb{Z}$. If $p \in \mathbb{N}$ is prime, let $\mathbb{F}_p$ denote the field with p elements. A ring is denoted by $\mathcal{R}$, and we implicitly assume $\mathcal{R}$ to be a commutative unit ring. Moreover, we implicitly assume all logarithms to be binary and denote them by log.

Bold upper case letters denote matrices whereas bold lower case letters denote row vectors. Column vectors are given as the transposed of row vectors, namely, for a row vector $\mathbf{v}$ the corresponding column vector is $\mathbf{v}^T$. The ordinary inner product of two vectors $\mathbf{v}$ and $\mathbf{w}$ is denoted by $(\mathbf{v}, \mathbf{w})$.

A set of l vectors $\mathbf{v_1}, \ldots, \mathbf{v_l}$ is called linearly independent in some ring $\mathcal{R}$ iff the only $\mathcal{R}$-linear combination of these vectors adding up to the zero vector is the zero linear combination. If a set of vectors is not linearly independent, it is called linearly dependent.

For $s, t \in \mathbb{N}$ the matrix $\mathbf{0^{s\times t}}$ describes the $s \times t$ zero matrix. For any positive integer n the $n \times n$ identity matrix is given by $\mathbf{I_n}$. The i-th unit vector is called $\mathbf{e}^i$.

For a vector $\mathbf{v}$ the value $(\mathbf{v})_i$ denotes the i-th entry of $\mathbf{v}$. For a matrix $\mathbf{M}$ the value $(\mathbf{M})_{ij}$ denotes the entry in the i-th row and in the j-th column. Note that these values are elements of $\mathcal{R}$. The value $\mathbf{M}_{i,\cdot}$ is the i-th row vector and $\mathbf{M}_{\cdot,j}$ the j-th column vector of $\mathbf{M}$. By $\mathbf{M}_{-i,\cdot}$, we denote the matrix constructed from $\mathbf{M}$ by deleting the i-th row. Analogously, $\mathbf{M}_{\cdot,-j}$ is constructed by deleting the j-th column of $\mathbf{M}$ and $\mathbf{M}_{-i,-j}$ by deleting the i-th row and the j-th column.

Given a matrix $\mathbf{M} \in \mathbb{Z}^{s\times t}$, a row vector $\mathbf{r} \in \mathbb{Z}^{1\times(t+1)}$ and a column vector $\mathbf{c}^T \in \mathbb{Z}^{(s+1)\times 1}$

and two indices i, j such that $(\mathbf{r})_j = (\mathbf{c})_i$, let $\mathbf{M}_{+i(\mathbf{r}),+j(\mathbf{c})^T}$ be the matrix

$$\mathbf{M}_{+i(\mathbf{r}),+j(\mathbf{c})} := \begin{pmatrix} (\mathbf{M})_{11} & \cdots & (\mathbf{M})_{1(j-1)} & (\mathbf{c})_1 & (\mathbf{M})_{1j} & \cdots & (\mathbf{M})_{1t} \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ (\mathbf{M})_{(i-1)1} & \cdots & (\mathbf{M})_{(i-1)(j-1)} & (\mathbf{c})_{i-1} & (\mathbf{M})_{(i-1)j} & \cdots & (\mathbf{M})_{(i-1)t} \\ (\mathbf{r})_1 & \cdots & (\mathbf{r})_{j-1} & (\mathbf{r})_j & (\mathbf{r})_{j+1} & \cdots & (\mathbf{r})_{t+1} \\ (\mathbf{M})_{i1} & \cdots & (\mathbf{M})_{i(j-1)} & (\mathbf{c})_{i+1} & (\mathbf{M})_{ij} & \cdots & (\mathbf{M})_{it} \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ (\mathbf{M})_{s1} & \cdots & (\mathbf{M})_{s(j-1)} & (\mathbf{c})_{s+1} & (\mathbf{M})_{sj} & \cdots & (\mathbf{M})_{st} \end{pmatrix}.$$

That is, we get the new matrix by inserting $\mathbf{r}$ as i-th row and $\mathbf{c}^T$ as j-th column into $\mathbf{M}$. The matrices defined by inserting only a row vector $\mathbf{r} \in \mathbb{Z}^{1\times t}$ or a column vector $\mathbf{c}^T \in \mathbb{Z}^{s\times 1}$ are denoted by $\mathbf{M}_{+i(\mathbf{r}),\cdot}$ and $\mathbf{M}_{\cdot,+j(\mathbf{c})}$ respectively.
Apart from these general operations on matrices, we will need some special matrices and their properties. Therefore, we will quote some theorems here. For more details regard [SW99].
Let $\mathbf{P}$ always denote a permutation matrix, that is, in each row/column of $\mathbf{P}$ there is exactly one component of value one, whereas all other components are zero.
Let $\mathbf{A} := \mathbf{A}(x, a, b)$ be the matrix such that $\mathbf{AM}$ describes the matrix derived from $\mathbf{M}$ by addition of x times row b to row a in $\mathbf{M}$. The addition of x times column b to column a in $\mathbf{M}$ is then done by multiplying $\mathbf{M}$ with $\mathbf{A}$ from the right. Matrices of this form are called elementary matrices.
Let $\mathbf{P}_{(ij)}$ be the permutation matrix which represents swapping the i-th and j-th row of a matrix $\mathbf{M}$ when multiplied to it from the left. Remark that

$$\mathbf{P}_{(ij)} = \mathbf{A}(1, j, i)\mathbf{A}(-1, i, j)\mathbf{A}(1, j, i)\mathbf{D}_j\,,$$

where $\mathbf{D}_j$ is a diagonal matrix with -1 on the j-th position and 1 on any other position. Thus, admitting a small change in the sign any permutation can be expressed via elementary matrices.
The matrices $\mathbf{A}$, $\mathbf{A}^T$ and $\mathbf{P}$ are unimodular, i. e. their determinant is of absolute value 1. Note that the product of two unimodular matrices is a unimodular matrix as well.
Having introduced the notation necessary, we can now cite the first theorem.

Theorem 2.1.1 ([SW99], Theorem 8.C.11)
Let $s, t \in \mathbb{N}$ and $\mathbf{M} \in \mathbb{Z}^{s\times t}$ be an $s \times t$-matrix of rank r with coefficients in $\mathbb{Z}$ and let $k := \min\{s, t\}$. Then there are elementary matrices $\mathbf{U}_1, \ldots, \mathbf{U}_p \in \mathbb{Z}^{s\times s}$ and $\mathbf{V}_1, \ldots, \mathbf{V}_q \in \mathbb{Z}^{t\times t}$ such that $\mathbf{U}_p \cdots \mathbf{U}_1\mathbf{M}\mathbf{V}_1 \cdots \mathbf{V}_q$ is a diagonal matrix $\mathbf{D} = \mathit{Diag}(a_1, \ldots, a_k) \in \mathbb{Z}^{s\times t}$, where $a_i|a_{i+1}$ for $i = 1, \ldots, r-1$ and $a_{r+1} = \ldots = a_k = 0$.

This theorem brings up a new definition.

Definition 2.1.2
The numbers $a_1, \ldots, a_r$ of Theorem 2.1.1 are called ***elementary divisors****.*

Note that the elementary divisors are uniquely determined except for the sign. For notational convenience throughout this thesis we assume all elementary divisors to be positive. The proof of Theorem 2.1.1 in [SW99] already gives the construction of the elementary divisors. As we are mainly interested in conditions on which all elementary divisors of a matrix are one, we will not give a proof of the general theorem here. Instead, we will prove a variant of this theorem which is adapted to our needs. Therefore, we will permute the elementary divisors to the last rows.
Further, note that in case of matrices with entries from a field, we can conclude easily that if $\mathbf{M}$ is of rank $k = t$, then the transformation works without $\mathbf{V}_1, \dots, \mathbf{V}_q$. We include a similar observation for matrices with values in $\mathbb{Z}$ in our theorem. We will make use of the following theorem in Section 5.1.

Theorem 2.1.3
Let $m \in \mathbb{N}_0, n \in \mathbb{N}$ and $\mathbf{M} \in \mathbb{Z}^{(m+n)\times n}$, rank$(\mathbf{M}) = n$. Then

$$\textit{All elementary divisors of } \mathbf{M} \textit{ are equal to } 1 \;\Leftrightarrow$$

$$\textit{There exists a unimodular matrix } \mathbf{U} \in \mathbb{Z}^{(m+n)\times(m+n)} \textit{ such that } \mathbf{UM} = \begin{pmatrix} \mathbf{0^{m\times n}} \\ \mathbf{I_n} \end{pmatrix}.$$

In order to prove Theorem 2.1.3 we will use the two following lemmata. Recall that the greatest common divisor of a vector is defined as the greatest common divisor of all its components.

Lemma 2.1.4
Let $r, t \in \mathbb{N}$, $t \le r$ and $\mathbf{v}^T \in \mathbb{Z}^{r\times 1}$ with $\gcd(\mathbf{v}) = 1$. *Then there exists a unimodular transformation $\mathbf{U} \in \mathbb{Z}^{r\times r}$ such that $\mathbf{U}\mathbf{v}^T = (\mathbf{e}^t)^T$.*

Proof: First, remark that $\gcd(\mathbf{v}) = \gcd((\mathbf{v})_1, \dots, (\mathbf{v})_r) = \gcd(\gcd((\mathbf{v})_1, \dots, (\mathbf{v})_{r-1}), (\mathbf{v})_r)$ $= \gcd(\dots \gcd((\mathbf{v})_1, (\mathbf{v})_2), \dots, (\mathbf{v})_r)$. The basic idea of the proof is to use the Euclidean Algorithm to iteratively compute the greatest common divisors. Each step of the Euclidean Algorithm induces a unimodular transformation which can be performed on $\mathbf{v}$. In the end, we can combine these transformations and obtain $\mathbf{U}$. As the greatest common divisor is one, $\mathbf{U}\mathbf{v}^T$ then denotes a unit vector.
Now let us look at the single steps in more detail. First, we compute the greatest common divisor of $(\mathbf{v})_1$ and $(\mathbf{v})_2$. We assume $(\mathbf{v})_1 \ge (\mathbf{v})_2$. If not, set $\mathbf{U_{20}} := \mathbf{P}_{(12)}$ and regard $(\mathbf{v^{20}})^T := \mathbf{U_{20}}\mathbf{v}^T$. In the first step of the Euclidean Algorithm, we divide $(\mathbf{v^{20}})_1$ by $(\mathbf{v^{20}})_2$ with remainder. That is, we compute s_1 and $r_1 \in \mathbb{Z}$ such that $(\mathbf{v^{20}})_1 = s_1(\mathbf{v^{20}})_2 + r_1$. The inputs to the next step are $(\mathbf{v^{20}})_2$ and $r_1 = (\mathbf{v^{20}})_1 \pmod{(\mathbf{v^{20}})_2}$. To transform $\mathbf{v^{20}}$ accordingly, set $\mathbf{U^{21}} := \mathbf{P}_{(12)}\mathbf{A}(-s_1, 1, 2)$. Then $(\mathbf{v^{21}})^T := \mathbf{U^{21}}(\mathbf{v^{20}})^T$ is a column vector with first entry $(\mathbf{v^{20}})_2$, second entry r_1 and all other entries equal to the corresponding entries in $\mathbf{v}$. Iterating this process until $r_{k_1} = 0$ (a situation which we reach due to the construction of the Euclidean Algorithm), we get

$$\underbrace{\mathbf{U^{2k_1}} \cdot \dots \cdot \mathbf{U^{20}}}_{=:\mathbf{U^2}} \mathbf{v}^T = (\gcd((\mathbf{v})_1, (\mathbf{v})_2), 0, (\mathbf{v})_3, \dots, (\mathbf{v})_r)^T =: (\mathbf{v^3})^T .$$

We then iterate the process for $i = 3, \dots, r$ performing the Euclidean Algorithm on the first and the i-th value of $\mathbf{v^i}$ and determining the unimodular transformations analogously. In the end we obtain a unimodular transformation $\tilde{\mathbf{U}} = \mathbf{U^{rk_r}} \cdot \ldots \cdot \mathbf{U^{20}}$ such that $\tilde{\mathbf{U}}\mathbf{v}^T = \gcd(\mathbf{v})\,(\mathbf{e}^1)^T$. As $\gcd(\mathbf{v}) = 1$ with a final permutation we obtain $\underbrace{\mathbf{P}_{(1t)}\tilde{\mathbf{U}}}_{=:\mathbf{U}}\mathbf{v}^T = (\mathbf{e}^t)^T$. All the matrices forming $\mathbf{U}$ are unimodular matrices. Thus, $\mathbf{U}$ is unimodular. This implies the claim. ■

Note that in actual computations the method described in the above proof often contains superfluous steps. Whenever a greatest common divisor of 1 is obtained, the value 1 can directly be used to eliminate all further entries of $\mathbf{v}$. The unit vector is, thus, obtained faster. This may already happen after one execution of the Euclidean algorithm if $\gcd((\mathbf{v})_1, (\mathbf{v})_2) = 1$.
Now we would like to extend our analysis from one single column to a matrix. Nevertheless, in the following lemma we take only knowledge on the greatest common divisors of columns into account as this is information which is easy to get from any matrix.

Lemma 2.1.5
Let $n \in \mathbb{N}$, $m \in \mathbb{N}_0$. Let $\mathbf{M} \in \mathbb{Z}^{(m+n)\times n}$. If $g_j(\mathbf{M}) := \gcd(\mathbf{M}_{\cdot,j}) > 1$ for all $j = 1, \dots, n$, then either $\mathbf{M}$ has at least one elementary divisor which is not equal to 1 or the number of elementary divisors of $\mathbf{M}$ is smaller than n.

Proof: The lemma holds for all singular matrices $\mathbf{M}$: If $\mathbf{M}$ is singular, then at least one column is linearly dependent on the other columns. This column can be eliminated by multiplication with a unimodular matrix $\tilde{\mathbf{V}}$ from the right. Thus, $\mathbf{M}\tilde{\mathbf{V}}$ contains a zero column. We permute this column to the last one. Then performing the elementary divisor algorithm on $\mathbf{M}\tilde{\mathbf{V}}$ ignoring the last column will result in a matrix having only entries on the diagonal. This is due to the elementary divisor condition of Theorem 2.1.1. The last diagonal entry of $\mathbf{U}\mathbf{M}\tilde{\mathbf{V}}\mathbf{V}$ is equal to zero. Thus, we have less than n elementary divisors. Therefore, we now restrict ourselves to matrices of full rank. We prove the lemma for regular matrices by induction on n.

First let $\mathbf{n = 1}$: Let $\mathbf{M} \in \mathbb{Z}^{(m+1)\times 1}$ with $g_1(\mathbf{M}) = \gcd(\mathbf{M}_{\cdot,1}) > 1$. Then, as $\mathbf{M} = \mathbf{M}_{\cdot,1}$, $g_1(\mathbf{M})$ divides any component of $\mathbf{M}$. Any entry in a matrix which can be constructed by unimodular transformations of $\mathbf{M}$ is a linear combination of entries of $\mathbf{M}$ and, thus, is a multiple of $g_1(\mathbf{M})$ as well. The same holds for the elementary divisor. This concludes the proof in the case of $n = 1$.

The **hypothesis** of the induction is as follows: For an $n \in \mathbb{N}$ and an arbitrary $m \in \mathbb{N}_0$ let $\mathbf{M} \in \mathbb{Z}^{(m+n)\times n}$ be a full rank matrix such that $g_j(\mathbf{M}) := \gcd(\mathbf{M}_{\cdot,j}) > 1$ for all $j = 1, \dots, n$. Then $\mathbf{M}$ has at least one elementary divisor which is not equal to 1.

For the **inductive step** we consider two cases.
First, let $n \in \mathbb{N}$, $\mathbf{M} \in \mathbb{Z}^{(m+n+1)\times(n+1)}$ such that $g_j(\mathbf{M}) := \gcd(\mathbf{M}_{\cdot,j}) > 1$ for all $j = 1, \dots, n+1$. Then if $g(\mathbf{M}) := \gcd(g_1(\mathbf{M}), \dots, g_{n+1}(\mathbf{M})) > 1$, all entries of $\mathbf{M}$ are multiples

of $g(\mathbf{M})$. Thus, all linear combinations of entries of $\mathbf{M}$ including the elementary divisors are multiples of $g(\mathbf{M})$ and, therefore, not equal to 1 and we are done.
Thus, we only have to consider the case of $g(\mathbf{M}) := \gcd(g_1(\mathbf{M}), \ldots, g_{n+1}(\mathbf{M})) = 1$. We start by performing the elementary divisor algorithm. That is, we apply the algorithm which gives the construction of Theorem 2.1.1. Here it is not important how the algorithm works exactly. We just have to know that in each step $\mathbf{M}$ is either multiplied from the left with a unimodular matrix $\mathbf{U}$ or from the right with a unimodular matrix $\mathbf{V}$.
Multiplications from the left only perform row operations on $\mathbf{M}$. Consequently, all elements in the j-th column of $\mathbf{UM}$ are divisible by $g_j(\mathbf{M})$ as they are linear combinations of elements of $\mathbf{M}_{\cdot,j}$ which are divisible by $g_j(\mathbf{M})$. Therefore, the value of $g_j(\mathbf{M})$ cannot be decreased by these operations.
Multiplications from the right are either column permutations or additions of multiples of one column to another. As permutations of columns only permute the greatest common divisors, it is sufficient to consider additions of multiples of one column to another. We perform only one such addition in one step. This implies that only the greatest common divisor of one column may be changed. We continue the process of transformations, using other transformations in between as the algorithm requires until the greatest common divisor $g_j(\mathbf{M})$ of one column becomes 1. (This will happen as $g(\mathbf{M}) := \gcd(g_1(\mathbf{M}), \ldots, g_{n+1}(\mathbf{M})) = 1$ and, thus, not all elements in the matrix share the same common divisor.) We then permute the j-th column to the first position. Let $\mathbf{U_1}$ and $\mathbf{V_1}$ be the unimodular transformations used so far. Let $\mathbf{M_1} := \mathbf{U_1 M V_1}$. Then $g_1(\mathbf{M_1}) = 1$ and $g_j(\mathbf{M_1}) > 1$ for $j \geq 2$. By further row operations using Lemma 2.1.4 we get a unimodular transformation $\mathbf{U_2}$ such that $\mathbf{U_2 M_1} = \left((\mathbf{e^1})^T \quad \mathbf{M_2} \right)$ for some $\mathbf{M_2} \in \mathbb{Z}^{(m+n+1)\times n}$ with $g_j(\mathbf{M_2}) = g_{j+1}(\mathbf{M_1}) > 1$. Now adding $-(\mathbf{M_2})_{1(j-1)}$-times the first column to the j-th only changes the component in the first row as all other entries in the first column are equal to zero. Let $\mathbf{V(j)}$ be the unimodular matrix performing this operation. Let $\mathbf{V_2} = \prod_{j=2}^{n+1} \mathbf{V(j)}$. Then

$$\left((\mathbf{e^1})^T \quad \mathbf{M_2} \right) \mathbf{V_2} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & & & \\ \vdots & & \mathbf{M_3} & \\ 0 & & & \end{pmatrix}.$$

As $(\mathbf{M_3})_{\cdot,j} = ((\mathbf{M_2})_{\cdot,j})_{-1,\cdot}$ we have $g_j(\mathbf{M_2}) | g_j(\mathbf{M_3})$ for all $j = 1, \ldots, n$. Moreover, $\mathbf{M_3}$ is of full rank n as $\mathbf{M_1}$ was.
This implies that $\mathbf{M_3} \in \mathbb{Z}^{(m+n)\times n}$ fulfills the conditions of the hypothesis. Thus, there are unimodular transformations $\mathbf{U_3} \in \mathbb{Z}^{(m+n)\times(m+n)}$ and $\mathbf{V_3} \in \mathbb{Z}^{n\times n}$ such that $\mathbf{M_4} := \mathbf{U_3 M_3 V_3}$ is a diagonal matrix with at least one diagonal element not equal to 1. Let

$$\mathbf{U_4} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & & & \\ \vdots & & \mathbf{U_3} & \\ 0 & & & \end{pmatrix} \text{ and } \mathbf{V_4} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & & & \\ \vdots & & \mathbf{V_3} & \\ 0 & & & \end{pmatrix}.$$

Then

$$\begin{aligned}\mathbf{U_4U_2U_1MV_1V_2V_4} &= \mathbf{U_4U_2M_1V_2V_4}\\ &= \begin{pmatrix} 1 & 0 & \cdots & 0\\ 0 & & & \\ \vdots & & \mathbf{U_3} & \\ 0 & & & \end{pmatrix}\begin{pmatrix} 1 & 0 & \cdots & 0\\ 0 & & & \\ \vdots & & \mathbf{M_3} & \\ 0 & & & \end{pmatrix}\begin{pmatrix} 1 & 0 & \cdots & 0\\ 0 & & & \\ \vdots & & \mathbf{V_3} & \\ 0 & & & \end{pmatrix}\\ &= \begin{pmatrix} 1 & 0 & \cdots & 0\\ 0 & & & \\ \vdots & & \mathbf{U_3M_3V_3} & \\ 0 & & & \end{pmatrix} = \begin{pmatrix} 1 & 0 & \cdots & 0\\ 0 & & & \\ \vdots & & \mathbf{M_4} & \\ 0 & & & \end{pmatrix}\end{aligned}$$

is a diagonal matrix the components of which are not all equal to 1. This implies that the elementary divisors of $\mathbf{M}$ do not all have the value 1 and the lemma is proven. ■

Now we can prove the main theorem.

Proof of Theorem 2.1.3: In a first step we prove the necessary condition.
Suppose there exists a unimodular matrix $\mathbf{U} \in \mathbb{Z}^{(m+n)\times(m+n)}$ such that $\mathbf{UM} = \begin{pmatrix} \mathbf{0^{m\times n}} \\ \mathbf{I_n} \end{pmatrix}$.
Let $\mathbf{V} := \mathbf{I_n}$ and $\mathbf{P} \in \mathbb{Z}^{(m+n)\times(m+n)}$ be defined as $\mathbf{P} = \mathbf{P}_{(n(m+n))} \cdot \ldots \cdot \mathbf{P}_{(1(m+1))}$. Namely, $\mathbf{P}$ is a permutation matrix which permutes the n last rows to the first n positions. Both $\mathbf{U}$ and $\mathbf{P}$ are unimodular. Consequently, $\mathbf{PU}$ is unimodular. The matrix $\mathbf{V}$ is unimodular as well. It is

$$(\mathbf{PU})\mathbf{MV} = \mathbf{P}\begin{pmatrix} \mathbf{0^{m\times n}} \\ \mathbf{I_n} \end{pmatrix}\mathbf{I_n} = \begin{pmatrix} \mathbf{I_n} \\ \mathbf{0^{m\times n}} \end{pmatrix}.$$

This implies that the elementary divisors of $\mathbf{M}$ are all equal to 1.

We prove the sufficient condition by induction on n for arbitrary $m \in \mathbb{N}_0$. Suppose all elementary divisors of $\mathbf{M}$ are equal to 1.
For a better understanding we start with the cases of $n = 1$ and $n = 2$.
Let $\mathbf{n = 1}$: Then $\mathbf{M} = \mathbf{M}_{\cdot,\mathbf{1}}$ is one column vector with $g_1(\mathbf{M}) = \gcd(\mathbf{M}_{\cdot,\mathbf{1}}) = 1$. (If $g_1(\mathbf{M}) > 1$, we directly get a contradiction by Lemma 2.1.5.)
By Lemma 2.1.4 there exists a unimodular transformation $\mathbf{U} \in \mathbb{Z}^{(m+1)\times(m+1)}$ such that $\mathbf{UM} = \mathbf{UM}_{\cdot,1} = (\mathbf{e^{m+1}})^T$. This implies that the case of $n = 1$ is correct.
Now let $\mathbf{n = 2}$: We divide this case into further subcases.

1. Suppose there exists a column $\mathbf{M}_{\cdot,j}$, $j \in \{1,2\}$, with $g_j(\mathbf{M}) = \gcd(\mathbf{M}_{\cdot,j}) = 1$. Analogous to the case of one column, there exists a unimodular transformation $\mathbf{U_1} \in \mathbb{Z}^{(m+2)\times(m+2)}$ such that $\mathbf{U_1M}_{\cdot,j} = \left(\mathbf{e^{m+j}}\right)^T$. Let $\mathbf{M_1} = \mathbf{U_1M}$. Then $\mathbf{M_1}$ consists of two columns, the j-th column which is equal to $\left(\mathbf{e^{m+j}}\right)^T$ and the $(3-j)$-th column with unknown components. Then $\mathbf{M_2} = (\mathbf{M_1})_{-(m+j),-j}$ is a one column matrix and $g_1(\mathbf{M_2}) = \gcd(\mathbf{M_2}_{\cdot,1})$. We again distinguish two subcases.

(a) $g_1(\mathbf{M_2}) = 1$:
Then a unimodular transformation $\mathbf{U_2} \in \mathbb{Z}^{(m+1)\times(m+1)}$ exists by Lemma 2.1.4 such that $\mathbf{U_2M_2} = (\mathbf{e^{m+1}})^T$. We define $\mathbf{U_3} := (\mathbf{U_2})_{+(m+j)(\mathbf{e^{m+j}}),+(m+j)(\mathbf{e^{m+j}})^T}$. That is, $\mathbf{U_3}$ is a matrix operating like $\mathbf{U_2}$ on $\mathbf{M_2}$ but leaving the rest of $\mathbf{M_1}$ unchanged. Let $\mathbf{U_3M_1} = \mathbf{U_3U_1M} =: \mathbf{M_3}$. The matrix $\mathbf{M_3}$ consists of two columns $\left(\mathbf{e^{m+j}}\right)^T$ and $(\mathbf{e_{m+3-j}})^T + \mathbf{c}^T$, where $\mathbf{c}^T \in \mathbb{Z}^{(m+2)\times 1}$ with $\mathbf{c}^T_{m+j,1} = x$ for an unknown $x \in \mathbb{Z}$ and $\mathbf{c}^T_{k,1} = 0$ for all $k \neq m+j$. That means, $\mathbf{M_3}$ is nearly of the required shape except for an integer in the $(m+j)$-th row and $(3-j)$-th column. Using the newly constructed 1 in the $(3-j)$-th column, which is the only entry in its corresponding row, we can now eliminate this value x as well. Let $\mathbf{A} := \mathbf{A(-x, m+j, m+3-j)}$ be the matrix such that $\mathbf{AM}$ describes the addition of $-x$ times row $m+3-j$ to row $m+j$ in $\mathbf{M}$. Then $\mathbf{AU_3U_1M} = \mathbf{AM_3} = \begin{pmatrix} \mathbf{0^{m\times 2}} \\ \mathbf{I_2} \end{pmatrix}$. Here, $\mathbf{AU_3U_1}$ is a unimodular transformation as the consisting matrices are unimodular. Consequently, the theorem holds in this case.

(b) $g_1(\mathbf{M_2}) > 1$:
By Lemma 2.1.5 we get that $\mathbf{M_2}$ has an elementary divisor greater than 1 and, therefore, the same holds for $\mathbf{M}$. This contradicts the preconditions.

2. Suppose that $g_1(\mathbf{M}), g_2(\mathbf{M}) \neq 1$.
Then we directly get a contradiction by Lemma 2.1.5.

Combining both subcases concludes the proof in the case of $n = 2$.

The **hypothesis** of our induction is as follows:
For an $n \in \mathbb{N}$, all $m \in \mathbb{N}_0$, let $\mathbf{M} \in \mathbb{Z}^{(m+n)\times n}$ be a matrix of which all n elementary divisors are equal to 1. Then there exists a unimodular transformation $\mathbf{U} \in \mathbb{Z}^{(m+n)\times(m+n)}$ such that

$$\mathbf{UM} = \begin{pmatrix} \mathbf{0^{m\times n}} \\ \mathbf{I_n} \end{pmatrix}. \tag{2.1}$$

Now let $n \in \mathbb{N}$, $\mathbf{M} \in \mathbb{Z}^{(m+n+1)\times(n+1)}$ with $n+1$ elementary divisors of value 1. We proceed analogously to the case of $n = 2$ and distinguish two cases.

1. There is $j \in \{1, \dots, n+1\}$ such that $g_j(\mathbf{M}) := \gcd(\mathbf{M}_{\cdot,j}) = 1$:
Then by Lemma 2.1.4 there is a unimodular matrix $\mathbf{U_1}$ such that $\mathbf{U_1M}_{\cdot,j} = \left(\mathbf{e^{m+j}}\right)^T$. Let $\mathbf{M_1} := \mathbf{U_1M}$. This implies that the first elementary divisor of $\mathbf{M}$ equals 1. Let $\mathbf{M_2} := (\mathbf{M_1})_{-(m+j),-j}$. Then $\mathbf{M_2}$ has n elementary divisors which are the elementary divisors of $\mathbf{M}$ except for the elementary divisor already computed. Consequently, all n elementary divisors of $\mathbf{M_2}$ are of value 1. Then by the hypothesis of the induction (2.1) there exists a unimodular matrix $\mathbf{U_2}$ such that $\mathbf{U_2M_2} = \begin{pmatrix} \mathbf{0^{m\times n}} \\ \mathbf{I_n} \end{pmatrix} =: \mathbf{M_3}$.

Let $\mathbf{U_3} := (\mathbf{U_2})_{+(m+j)(\mathbf{e^{m+j}}),+(m+j)(\mathbf{e^{m+j}})^T}$. Then

$$\begin{aligned}\mathbf{U_3U_1M} &= \mathbf{U_3M_1}\\ &= (\mathbf{U_2})_{+(m+j)(\mathbf{e^{m+j}}),+(m+j)(\mathbf{e^{m+j}})^T}(\mathbf{M_2})_{+(m+j)(\mathbf{c}),+j(\mathbf{e^{m+j}})^T}\\ &= (\mathbf{M_3})_{+(m+j)(\mathbf{c}),+j(\mathbf{e^{m+j}})^T}\,.\end{aligned}$$

Here $\mathbf{c} := (\mathbf{M_1})_{m+j,\cdot}$. Consequently, $\mathbf{c}_j = 1$. In the last step, we want to eliminate all non-zero entries in the $(m+j)$-th row except for the one in the j-th column. To do so, we subtract $\mathbf{c}_i$ times the $(m+i)$-th from the $(m+j)$-th row, for $i = 1, \ldots, j-1, j+1, \ldots, m+n$. Let $\mathbf{A}$ denote the unimodular transformation corresponding to these operations. Then

$$\mathbf{A}(\mathbf{M_3})_{+(m+j)(\mathbf{c}),+j(\mathbf{e^{m+j}})^T} = \begin{pmatrix} \mathbf{0^{m\times(n+1)}} \\ \mathbf{I_{n+1}} \end{pmatrix}.$$

Thus, the theorem is proven.

2. For all $j \in \{1, \ldots, n+1\}$ it holds that $g_j(\mathbf{M}) := \gcd(\mathbf{M}_{\cdot,j}) > 1$:
Then we get a contradiction by Lemma 2.1.5. ■

2.2 Algebraic Methods

In this section basic notation, definitions and constructions from algebra are introduced. This thesis deals with problems that occur in the context of cryptography. An example is the problem of factoring: Given a composite integer $N = pq$, determine its factors p and q. This as well as variants of it will be introduced in the subsequent chapters. Hence, we need tools to describe such problems. First, we introduce polynomial rings. For more details consider for example [CLO97].

Definition 2.2.1
*A **monomial** in $x_1, \ldots, x_l$ is a product $x_1^{i_1}x_2^{i_2}\cdot\ldots\cdot x_l^{i_l}$ with $i_1, \ldots, i_l \in \mathbb{N}_0$. The value $\deg(x_1^{i_1}x_2^{i_2}\cdot\ldots\cdot x_l^{i_l}) = i_1 + \ldots + i_l$ is called the **total degree** of $x_1^{i_1}x_2^{i_2}\cdot\ldots\cdot x_l^{i_l}$, whereas $\deg_{x_j}(x_1^{i_1}x_2^{i_2}\cdot\ldots\cdot x_l^{i_l}) = i_j$ is called the **degree in** x_j. Let $\mathcal{R}$ be a ring. An $\mathcal{R}$-linear combination of monomials $f(x_1, \ldots, x_l) = \sum_{(i_1,\ldots,i_l)} a_{(i_1,\ldots,i_l)}x_1^{i_1}\ldots x_l^{i_l}$ with $a_{(i_1,\ldots,i_l)} \in \mathcal{R}$ is called a **polynomial**. The set of all these polynomials is the **polynomial ring** $\mathcal{R}[x_1, \ldots, x_l]$.*

Using this notation, the problem of factoring can be interpreted as the problem of finding non-trivial solutions of the polynomial equation $f(x_1, x_2) := N - x_1x_2 = 0$. This way, additional information can be added easily to the description of a problem. Instead of describing a problem by a single polynomial equation, we describe it by a system of polynomial equations. Then a solution of the problem is given by a common solution of all of the equations.
In general, however, it is not clear if a new equation also contains new information. Two

equations $f_1(x_1,\dots,x_l) = 0$ and $f_2(x_1,\dots,x_l) = 0$ can either share the same set of solutions, share some solutions or the set of their solutions can be completely disjoint. In the two latter cases, we get additional information by taking two instead of one equation. However, in the first case taking two equations does not help to find the solution. We need a criterion to identify this case. We remark that two polynomials share exactly the same set of solutions if one of the polynomials is an integer multiple of the other. For systems of polynomials, this is captured by the following definition.

Definition 2.2.2
*Let $f_1(x_1,\dots,x_l),\dots,f_k(x_1,\dots,x_l)$ and $g(x_1,\dots,x_l) \in \mathcal{R}[x_1,\dots,x_l]$ be a set of polynomials. Let $\mathcal{F} := \{f_1(x_1,\dots,x_l),\dots,f_k(x_1,\dots,x_l)\}$. Then $\mathcal{F}$ is called **linearly independent** over $\mathcal{R}$ iff*

$$\sum_{i=1}^{k} a_i f_i(x_1,\dots,x_l) = 0 \text{ with } a_i \in \mathcal{R} \Rightarrow a_i = 0 \text{ for all } i = 1,\dots,k\,. \tag{2.2}$$

*If condition (2.2) does not hold, then $\mathcal{F}$ is called **linearly dependent** over $\mathcal{R}$.
The polynomial $g(x_1,\dots,x_l)$ is called **linearly independent** of $\mathcal{F}$ over $\mathcal{R}$ iff*

$$\text{there are no } a_i \in \mathcal{R} \text{ such that } \sum_{i=1}^{k} a_i f_i(x_1,\dots,x_l) = g\,. \tag{2.3}$$

*If such an $\mathcal{R}$-linear combination of elements of $\mathcal{F}$ exists, then $g(x_1,\dots,x_l)$ is called **linearly dependent** of $\mathcal{F}$ over $\mathcal{R}$.*

Note that if $\mathcal{R}$ is a field, the two notions of linear independence given above are equivalent. In a ring $\mathcal{R}$ which is not a field, however, a set of polynomials $\mathcal{F} \cup \{g\}$ can be linearly dependent whereas g still is linearly independent of $\mathcal{F}$ over $\mathcal{R}$. We will make use of this difference in Section 5.1.

Now we have found a criterion whether an additional equation gives additional information on the solutions. That is, if the additional equation is linearly independent of the previous ones, we obtain further information using it. Unfortunately, this does not tell anything whether we can really determine the solutions. A major interest of this thesis is solving such systems of equations. Special systems of equations which can be solved efficiently are systems of dimension zero. We are going to define these systems now. Solving them also forms an important step in the analyses in the method of Coppersmith which will be presented in Section 2.3.

Let us regard the following system of $k \in \mathbb{N}$ equations in $l \in \mathbb{N}$ variables over a field $\mathbb{F}$

$$\begin{aligned} f_1(x_1,\dots,x_l) &= 0 \\ &\vdots \\ f_k(x_1,\dots,x_l) &= 0\,. \end{aligned} \tag{2.4}$$

Definition 2.2.3
The system (2.4) is called ***system of dimension zero*** *iff it has a finite number of solutions in the algebraic closure $\bar{\mathbb{F}}$ of $\mathbb{F}$.*

In our analyses we will often encounter systems of multivariate polynomial equations in $\mathbb{Z}$, not in any field. For our analyses, however, we can consider them as equations over $\mathbb{Q}$. If the given system is zero dimensional when regarded as system in $\mathbb{Q}$, then the number of solutions (in $\bar{\mathbb{Q}}$) is finite. Thus, the number of solutions which are elements of $\mathbb{Z}$ is finite as well. They can be determined by determining all solutions in the algebraic closure of $\mathbb{Q}$ and then just taking the ones in $\mathbb{Z}$ which we are searching for.

In what follows we will discuss two methods how to determine the solutions of zero dimensional systems, namely, Groebner bases and resultants. First of all, we need to introduce some additional notation.
If we consider e. g. division algorithms operating on polynomials, we need to decide which monomials to consider first. That is, we need an ordering of monomials.

Definition 2.2.4
A ***monomial ordering*** *on a set of monomials $x^\alpha = \prod_{i=1}^{l} x_i^{\alpha_i}$, $\alpha_i \in \mathbb{N}_0$, is a relation $\leq$ on $\mathbb{N}_0^l$ satisfying:*

1. *$\leq$ is a total (or linear) ordering on $\mathbb{N}_0^l$, namely, it is reflexive ($\alpha \leq \alpha$), transitive ($\alpha \leq \beta, \beta \leq \gamma \Rightarrow \alpha \leq \gamma$) and antisymmetric ($\alpha \leq \beta \leq \alpha \Rightarrow \alpha = \beta$).*

2. *If $\alpha \leq \beta$ and $\gamma \in \mathbb{N}_0^l$, then $\alpha + \gamma \leq \beta + \gamma$.*

3. *$\leq$ is a well-ordering on $\mathbb{N}_0^l$, i. e. every non-empty subset of $\mathbb{N}_0^l$ has a smallest element with regard to $\leq$.*

We give some standard monomial orderings as well as other orderings which will come in useful later.

Example 2.2.5
Let $\alpha = (\alpha_1, \ldots, \alpha_l), \beta = (\beta_1, \ldots, \beta_l) \in \mathbb{N}_0^l$, $l \in \mathbb{N}$ and $|\alpha| := \sum_{i=1}^{l} \alpha_i$.

1. ***Lexicographical ordering***
   It is $\alpha >_{\text{lex}} \beta$ if the leftmost non-zero entry of $\alpha - \beta$ is positive, and $\alpha = \beta$ if $\alpha - \beta$ denotes the zero vector. This definition implies $x_1 > \ldots > x_l$. Allowing to permute the variables, that is setting $x^\alpha = \prod_{i=1}^{l} x_{\psi(i)}^{\alpha_i}$ with a permutation ψ, we obtain a different lexicographical ordering. Unless stated otherwise, we assume $x_1 > \ldots > x_l$.

2. ***Graded lexicographical ordering***
   It is $\alpha >_{\text{grlex}} \beta$ if $|\alpha| > |\beta|$ or $|\alpha| = |\beta|$ and $\alpha >_{\text{lex}} \beta$. Further, $\alpha =_{\text{grlex}} \beta$ if $\alpha =_{\text{lex}} \beta$.

3. ***Graded reverse lexicographical ordering***
   It is $\alpha >_{\text{grevlex}} \beta$ if $|\alpha| > |\beta|$ or $|\alpha| = |\beta|$ and in $\alpha - \beta \in \mathbb{Z}^l$ the rightmost non-zero entry is negative.

4. ***Inverse graded lexicographical ordering***
   It is $\alpha \geq_{\text{invgl}} \beta$ if $\alpha \leq_{\text{grlex}} \beta$. Note that although we call this construction "ordering", it actually is not an ordering according to our definition. We do not always have a smallest element in any non-empty set. However, we do have a largest element in any set as the graded lexicographical ordering fulfills the definition of a monomial ordering. Furthermore, when using this "ordering" we will only deal with finite sets of monomials. In those sets, we can determine a smallest element. This suffices for our applications.

With regard to monomial orderings one can define certain distinguished monomials and terms in polynomials.

Definition 2.2.6
For $\alpha \in \mathbb{N}_0^l$ and $x = (x_1, \ldots, x_l)$ let x^α denote $\prod_{i=1}^l x_i^{\alpha_i}$. Let $f(x_1, \ldots, x_l) = \sum_\alpha c_\alpha x^\alpha$ be a non-zero polynomial in $\mathcal{R}[x_1, \ldots, x_l]$. Let $\leq$ be a monomial ordering. Let $\bar{\alpha}$ denote the maximum of all α occurring in f.
*Then $LM(f) := x^{\bar{\alpha}}$ is called the **leading monomial** of f, and $LC(f) := c_{\bar{\alpha}}$ is called the **leading coefficient** of f. Together, $LT(f) := c_{\bar{\alpha}} x^{\bar{\alpha}} = LC(f) \cdot LM(f)$ denotes the **leading term** of f.*

So far we have defined systems of equations of dimension zero. The system itself is described by a set $\mathcal{F}$ of polynomials. We can add as many multiples hf, $h \in \mathcal{R}[x_1, \ldots, x_l]$ of polynomials $f \in \mathcal{F}$ to $\mathcal{F}$ as they have at least the same solutions as f and, therefore, do not change the set of common solutions. Furthermore, we can include any linear combination of elements of $\mathcal{F}$. By this, we get a more general description of the given system of equations. This is captured by the notion of an ideal.

Definition 2.2.7
*Let $I \subseteq \mathcal{R}[x_1, \ldots, x_l]$. Then I is called an **ideal** if*

1. *$0 \in I$,*
2. *If $f, g \in I$, then $f - g \in I$,*
3. *If $f \in I$ and $h \in \mathcal{R}[x_1, \ldots, x_l]$, then $hf \in I$.*

For a given set of polynomials $\{g_1, \ldots, g_k\}$ let $\langle g_1, \ldots g_k \rangle := \{\sum_{i=1}^k h_i g_i, h_i \in \mathcal{R}[x_1, \ldots, x_l]\}$ denote the ideal defined by these polynomials.
*Further, if for any $f, g \in \mathcal{R}[x_1, \ldots, x_l]$ the fact that $fg \in I \subset \mathcal{R}[x_1, \ldots, x_l]$ implies $f \in I$ or $g \in I$, then I is called a **prime ideal**.*

Note that the set $\{\sum_{f \in \mathcal{F}} h_f f \mid h_f \in \mathcal{R}[x_1, \ldots, x_l]\}$ forms an ideal. We call it the ideal induced by $\mathcal{F}$ and denote it by $\langle \mathcal{F} \rangle$. Thus, the solutions we are searching for correspond to the set of joint solutions of all polynomials in the ideal $\langle \mathcal{F} \rangle$.

Definition 2.2.8
Let $f_1, \dots, f_k$ be polynomials in $\mathcal{R}[x_1, \dots, x_l]$. Let $I \subseteq \mathcal{R}[x_1, \dots, x_l]$ be an ideal. Then the set

$$V(f_1, \dots, f_k) := \{(a_1, \dots a_l) \mid f_i(a_1, \dots, a_l) = 0 \text{ for all } 1 \leq i \leq k\}$$

is called the ***affine variety*** *of $f_1, \dots, f_k$. Analogously, the set*

$$V(I) := \{(a_1, \dots a_l) \mid f(a_1, \dots, a_l) = 0 \text{ for all } f \in I\}$$

is called the ***variety*** *of the ideal I.*

A variety is said to be of **dimension zero** if it comprises only finitely many elements. A variety of dimension zero, thus, corresponds to a system of dimension zero.
Starting with a system of dimension zero, let us have a look at how we can determine its solutions. One method makes use of resultants. We recall their definition and some basic properties here to see how they can be used.

Definition 2.2.9
Let $f_1(x_1, \dots, x_l) := \sum_{i=0}^{m} a_i(x_2, \dots, x_l)x_1^i$ and $f_2(x_1, \dots, x_l) := \sum_{i=0}^{n} b_i(x_2, \dots, x_l)x_1^i$ be two polynomials in $\mathcal{R}[x_1, \dots, x_l]$. Then the ***resultant*** *of f_1 and f_2 with respect to x_1 is defined as the determinant*

$$Res_{x_1}(f_1, f_2) := \det \left(\underbrace{\begin{matrix} a_m & & & \\ a_{m-1} & a_m & & \\ & a_{m-1} & \ddots & \\ \vdots & & \ddots & a_m \\ & \vdots & & a_{m-1} \\ a_0 & & & \\ & a_0 & & \vdots \\ & & \ddots & \\ & & & a_0 \end{matrix}}_{n \text{ columns}} \underbrace{\begin{matrix} b_n & & & \\ b_{n-1} & b_n & & \\ & b_{n-1} & \ddots & \\ \vdots & & \ddots & b_n \\ & \vdots & & b_{n-1} \\ b_0 & & & \\ & b_0 & & \vdots \\ & & \ddots & \\ & & & b_0 \end{matrix}}_{m \text{ columns}} \right) .$$

Remark that $a_i = a_i(x_2, \dots, x_l)$ and $b_j = b_j(x_2, \dots, x_l)$ denote polynomials in $\mathcal{R}[x_2, \dots, x_l]$.

The resultant is a polynomial in $\mathcal{R}[x_2, \dots, x_l]$, that is, a polynomial in only $l - 1$ variables. It has the following properties which we only quote from [CLO97].

Lemma 2.2.10 ([CLO97], Section 3.6)
Let $f_1(x_1, \dots, x_l)$ and $f_2(x_1, \dots, x_l) \in \mathcal{R}[x_1, \dots, x_l]$ be two polynomials with positive degree in x_1. Then

1. *$Res_{x_1}(f_1, f_2)$ is in the first elimination ideal $\langle f_1, f_2 \rangle \cap \mathcal{R}[x_2, \dots, x_l]$.*

2. *$\mathrm{Res}_{x_1}(f_1, f_2) = 0$ iff f_1 and f_2 have a common factor in $\mathcal{R}[x_1, \ldots, x_l]$ which has positive degree in x_1.*

3. *If $(\bar{x}_1, \ldots, \bar{x}_l)$ denotes a common zero of f_1 and f_2, then also*

$$\mathrm{Res}_{x_1}(f_1, f_2)(\bar{x}_2, \ldots, \bar{x}_l) = 0\,.$$

These properties already indicate how we can make use of resultants to solve a given system of l polynomial equations in $\mathcal{R}[x_1, \ldots, x_l]$. Whenever we combine two equations in l variables, we obtain a new equation in $l - 1$ variables. Therefore, if we combine all equations with the first one, we obtain $l-1$ new equations in $l-1$ variables. We can iterate this process until we have only one equation in one variable. If $\mathcal{R}$ denotes a field or the ring of integers $\mathbb{Z}$, which is the case in most of our applications, we can solve the univariate equation. Plugging the result into a bivariate equation constructed one step before, this equation becomes univariate. Then it can be solved as well. By successively substituting all the values which we have obtained, we can determine the solutions of the given system of equations. The following scheme illustrates the process of successively computing the resultants:

$$\left.\begin{array}{l} f_1(x_1, \ldots, x_l) \\ f_2(x_1, \ldots, x_l) \\ \vdots \\ f_l(x_1, \ldots, x_l) \end{array}\right\} \left.\begin{array}{l} \mathrm{Res}_{x_1}(f_1, f_2) =: f_{12} \\ \vdots \\ \mathrm{Res}_{x_1}(f_1, f_l) =: f_{1l} \end{array}\right\} \cdots \left.\vphantom{X}\right\} \mathrm{Res}_{x_{l-1}}(f_{(l-2)(l-1)}, f_{(l-2)l}) =: f_{(l-1)l}\,. \tag{2.5}$$

Problems during the computation might occur if one of the resultants becomes zero. This, however, violates the precondition that our system is of dimension zero.
Remark that we have just considered systems of l equations in $\mathcal{R}[x_1, \ldots, x_l]$. We can, of course, extend this method to systems of $k \in \mathbb{N}$ equations. Then it is $k \geq l$ as otherwise we would not have a system of dimension zero. Computations can be performed analogously. However, resultants with value zero may occur. This does not pose any problems as the system is zero dimensional. This ensures that there will be at least l non-zero polynomials in l variables. It might be necessary though to compute resultants using other pairs of polynomials.

Let us have a look at the running time of computing roots of polynomials by successively computing resultants. We again consider a zero dimensional system of l equations in l variables. Let δ denote the maximum total degree occurring in any of the equations. Thus, the maximum total degree of any of the coefficients of x_1^i in any of the f_j is also δ. Consequently, the resultant $\mathrm{Res}_{x_1}(f_1, f_j) = f_{1j}$ is a polynomial of total degree smaller than or equal to δ^2. Iterating this process, the total degree of $\mathrm{Res}_{x_{l-1}}(f_{(l-2)(l-1)}, f_{(l-2)l}) = f_{(l-1)l}$ is smaller than or equal to $\delta^{2^{l-1}}$. The major step in the resultant computation is the determinant calculation which has running time cubic in its dimension. That is, in the i-th iteration, the computation complexity is $\mathcal{O}((l - i)\delta^{3 \cdot 2^i})$. By some rough estimates the general running time of determining the solutions is then $\mathcal{O}(l^2 \delta^{3 \cdot 2^l})$. This is double exponential in l and, thus, not efficient for a non-constant number of variables.

Now we present the second method to solve systems of equations of dimension zero. This method is not based on the system of polynomials $\mathcal{F}$ itself but on the ideal $\langle \mathcal{F} \rangle$ defined by them. The same general ideal can be described in many different ways. Here we give a special description of an ideal. In what follows, we will see how this "good description" helps to determine solutions of the equations.

Definition 2.2.11
*Let I be an ideal. A finite subset $G = \{h_1, \dots, h_\kappa\}$ of I is called a **Groebner basis** if*

$$\langle LT(h_1), \dots, LT(h_\kappa) \rangle = \langle LT(I) \rangle \, .$$

*A minimal set with this property is called **minimal Groebner basis**.*

If not stated otherwise, we will compute Groebner bases with respect to lexicographical ordering. In the context of this work we often encounter systems of equations $S = \{g_1, \dots, g_\lambda\}$ which we would like to solve. When saying that we compute the Groebner basis of S, we implicitly assume to construct the ideal I of S and then to compute its Groebner basis. Groebner bases have several advantageous properties compared to arbitrary bases of ideals. Here we concentrate on those properties that help us to determine solutions of a system of equations. By a Groebner basis G of an ideal $I \subseteq \mathcal{R}[x_1, \dots, x_l]$ computed with respect to lexicographical ordering we directly get the Groebner bases of the ideals $I_j := I \cap \mathcal{R}[x_{j+1}, \dots, x_l]$, the so called j-th elimination ideals.

Theorem 2.2.12 (Elimination Theorem)
Let $I \subseteq \mathcal{R}[x_1, \dots, x_l]$ be an ideal and let G be a Groebner basis of I computed with respect to the lexicographical ordering such that $x_1 > \dots > x_l$. Then, for every $0 \le j < l$, the set

$$G_j := G \cap \mathcal{R}[x_{j+1}, \dots, x_l] \tag{2.6}$$

is a Groebner basis of the j-th elimination ideal $I_j := I \cap \mathcal{R}[x_{j+1}, \dots, x_l]$.

PROOF: To prove this, we have to show that for all $j = 0, \dots, l-1$ it is $\langle LT(G_j) \rangle = \langle LT(I_j) \rangle$.
Let $j \in \{0, \dots, l-1\}$ be fixed. As all polynomials in G_j are trivially also in I_j, it is $\langle LT(G_j) \rangle \subseteq \langle LT(I_j) \rangle$.
For the opposite inclusion let $f \in I_j$. To show that $LT(f) \in \langle LT(G_j) \rangle$ we have to show that $LT(h)$ divides $LT(f)$ for some $h \in G_j$. As $f \in I$ and G is a Groebner basis of I, we know that there is a polynomial $g \in G$ such that $LT(g)$ divides $LT(f)$. As $f \in I_j$, it follows that $LT(g)$ is a monomial in $\mathcal{R}[x_{j+1}, \dots, x_l]$. As G has been computed with respect to the lexicographical ordering with weights $x_1 > \dots > x_l$, any monomial which contains a power of x_i with $i \le j$ is greater than all monomials not involving any of these variables. Thus, all monomials of g are elements of $\mathcal{R}[x_{j+1}, \dots, x_l]$. As a consequence, $g \in \mathcal{R}[x_{j+1}, \dots, x_l]$ and, thus, $g \in G_j$. This concludes the proof. ■

The Elimination Theorem thereby helps us to structure the elements of an ideal I. If the ideal I contains a univariate polynomial in x_l, a univariate polynomial will also be contained in G. Analogously, if I contains a polynomial f in $x_{j+1}, \ldots, x_l$, then a polynomial in $x_{j+1}, \ldots, x_l$ is also part of G. On the assumption that for all $j = 0, \ldots, l-1$ the set $G_j \setminus G_{j+1}$ is not empty, the Groebner basis directly gives us a sequence of polynomials $g_1, \ldots, g_l$ such that g_j is a polynomial in $x_j, \ldots, x_l$. We can then calculate the solutions by solving a system of equations of the following structure:

$$\begin{aligned} g_1(x_1, \ldots, x_l) &= 0 \\ &\vdots \\ g_{l-1}(x_{l-1}, x_l) &= 0 \\ g_l(x_l) &= 0\,. \end{aligned} \tag{2.7}$$

This system of equations can be solved easily. First, we calculate the solution $\bar{x}_l$ of $g_l(x_l) = 0$. This is a univariate equation over the integers and can be solved efficiently by standard techniques like Newton iteration ([SB02], Chapter 5). Then we substitute x_l by $\bar{x}_l$ in g_{l-1} and obtain a univariate equation in x_{l-1}. We iterate this process until we have calculated all solutions as solutions of univariate equations. Hence, the problem can be solved efficiently.
To apply this method, however, we have to ensure that $G_j \setminus G_{j+1} \neq \emptyset$ for all values of j. This fact is implied by the zero dimensionality of $V(I)$.

There are various algorithms to compute Groebner bases of ideals which are adapted to different types of ideals or orderings. They differ with regard to efficiency. The original algorithm to compute a Groebner basis was given by Bruno Buchberger [Buc65, Buc06]. If $\mathcal{R}$ is Noetherian, i. e. in any ascending chain of ideals the ideals are equal from some point on, then by Buchberger's algorithm a Groebner basis is computed in finitely many steps. The algorithm, however, includes several superfluous computations. In 1993, Jean-Charles Faugère et al. developed an algorithm called FGLM to compute Groebner bases more efficiently [FGLM93]. With this algorithm it is further possible to switch between orders. Namely, the Groebner basis can be computed with respect to any order. A transformation is applied afterwards to get back to the lexicographical ordering we need in our application. This is helpful as Groebner bases with respect to graded orders often perform better.
In 1999, Jean-Charles Faugère presented a different algorithm, F4 [Fau99], which is again based on Buchberger's algorithm but leaves out several unnecessary computations. By this, the algorithm becomes more efficient. Some years later, Jean-Charles Faugère again improved on his algorithm. The new algorithm F5 [Fau02] eliminates all superfluous computations but imposes some additional constraints. These two algorithms are more adapted to graded reverse lexicographical ordering. They are, thus, more efficient but do not include the step of transforming a Groebner basis with respect to one ordering into a Groebner basis with respect to a different one. An elementary algorithm which provides exactly this step is the Groebner Walk by Stéphane Collart et al. [CKM97].

Unfortunately, the running time of none of the algorithms to compute a Groebner basis is completely understood. An upper bound on the calculation of a Groebner basis of some ideal $I = \langle f_1, \dots, f_k \rangle \subseteq \mathbb{F}[x_1, \dots, x_l]$ with $\deg(f_i) \leq \delta$ is given by $\delta^{2^{\mathcal{O}(l)}}$. This bound is proven to be strict in [MM82]. The worst case complexity of computing a Groebner basis is, thus, double exponential in the number of variables. For several other ideals of more practical interest better bounds like $\delta^{\mathcal{O}(l)}$ have been shown [Laz83].
Note that these bounds are only given for ideals in $\mathbb{F}[x_1, \dots, x_l]$ for a field $\mathbb{F}$. This, however, does not pose any problems. Although we mainly have polynomials with coefficients in $\mathbb{Z}$, we can take any field like $\mathbb{Q}$ comprising $\mathbb{Z}$ and perform the computations there. If the number of elements in the variety is finite, one can just choose the ones which are also elements of $\mathbb{Z}$ afterwards. Note that the set of solutions of a system of dimension zero can be described as a set of solutions of a system of the structure given in (2.7). Namely, by a system of equations in which each equation introduces a new variable. A univariate equation of degree δ has at most δ roots in $\bar{\mathbb{Q}}$. Consequently, the number of solutions in $\bar{\mathbb{Q}}$ is polynomial in the maximum total degree δ of polynomials in a Groebner basis. Therefore, for a constant number of variables, the correct solutions can be found efficiently. This is the case in all our applications.

Up to now we have considered systems of equations in $\mathcal{R}[x_1, \dots, x_l]$. That is, all the equations are elements of the same polynomial ring. This includes the cases $\mathcal{R} = \mathbb{Z}$ and $\mathcal{R} = \mathbb{Z}_N$ for some $N \in \mathbb{N}$. Namely, we can analyze systems of modular equations. In the context of this thesis, however, we will also encounter systems of modular equations with different moduli, such that $f_i(x_1, \dots, x_l) \equiv 0 \pmod{N_i}$, $i = 1, \dots k$, and $\gcd(N_i, N_j) = 1$ if $i \neq j$. We have to apply different techniques to determine the solutions of such a system. A possible way to combine these equations is by Chinese Remaindering which is described e. g. in [Hås88, Sho05]. The resulting polynomial can then be analyzed as a single polynomial in some polynomial ring.

Theorem 2.2.13 (Chinese Remainder Theorem)
Let $k \in \mathbb{N}$. Let $w \in \mathbb{N}$, $w > 1$. For $i = 1, \dots, k$ let $N_i \in \mathbb{N}$ be pairwise relatively prime numbers, and let $f_i(x_1, \dots, x_l) \in \mathbb{Z}_{N_i}[x_1, \dots, x_l]$ be polynomials such that at most w different monomials occur in any of the polynomials.
Then there exists a unique polynomial $f(x_1, \dots, x_l)$ modulo $M := \prod_{i=1}^{k} N_i$ such that

$$f(x_1, \dots, x_l) \equiv f_i(x_1, \dots, x_l) \pmod{N_i}. \tag{2.8}$$

The polynomial $f(x_1, \dots, x_l)$ can be determined in time $\mathcal{O}(w \log^2 M)$.

Proof: Let $M := \prod_{i=1}^{k} N_i$, $M_i := \frac{M}{N_i}$ and M_i' be the inverse of M_i modulo N_i for $i = 1, \dots, k$. The existence of such an inverse is guaranteed by $\gcd(M_i, N_i) = 1$. Then

$$f(x_1, \dots, x_l) := \sum_{i=1}^{k} M_i M_i' f_i(x_1, \dots, x_l)$$

is the desired polynomial. If we look at $f(x_1,\dots,x_l)$ modulo N_j for $j \in \{1,\dots,k\}$, all summands with index $i \neq j$ cancel out (as N_j divides M_i) and $M_j M'_j f_j(x_1,\dots,x_l) \equiv f_j(x_1,\dots,x_l) \pmod{N_j}$.
Now suppose that $g(x_1,\dots,x_l)$ is another polynomial fulfilling the required conditions. Then this implies $f(x_1,\dots,x_l) - g(x_1,\dots,x_l) \equiv 0 \pmod{N_i}$ for all $i = 1,\dots,k$, and, therefore, also $f(x_1,\dots,x_l) \equiv g(x_1,\dots,x_l) \pmod{M}$.
Multiplication modulo M and calculating the inverses by the Extended Euclidean Algorithm can be performed in time $\mathcal{O}(\log^2 M)$. Determining all coefficients of f then results in a running time of $\mathcal{O}(w \log^2 M)$ for the complete algorithm. ■

Note that in case of univariate polynomials $f_i(x) \in \mathbb{Z}_{N_i}[x]$ the maximum number of monomials w corresponds to the maximum degree δ of any of the polynomials. More precisely, we have $w \leq \delta + 1$.

2.3 Lattices

Problems derived from public key encryption or factoring can often be described as multivariate equations in the secret parameters, sometimes modular, sometimes over the integers. Equations of this type can be linearized by introducing new unknowns for any unknown monomial. If the unknowns are small enough, solutions can then be determined with methods used in the theory of lattices. Thus, we will give a brief introduction to lattices here. For a more thorough introduction we refer the reader to [MG02].
For our purposes it is sufficient to consider only integer lattices. However, analogous definitions can be made over the reals.

Definition 2.3.1
*Let $d, n \in \mathbb{N}$, $d \leq n$. Let $\mathbf{v_1},\dots,\mathbf{v_d} \in \mathbb{Z}^n$ be linearly independent vectors. Then the set of all integer linear combinations of the $\mathbf{v_i}$ spans an integer **lattice** L, i.e.*

$$L := \left\{ \sum_{i=1}^{d} a_i \mathbf{v_i} \mid a_i \in \mathbb{Z} \right\} .$$

*We call $\mathbf{B} = \begin{pmatrix} \mathbf{v_1} \\ \vdots \\ \mathbf{v_d} \end{pmatrix}$ a **basis matrix** of the lattice L, the value d denotes the **dimension** or **rank** of the lattice. The lattice is said to have **full rank** if $d = n$. The **determinant** $\det(L)$ of a lattice is the volume of the parallelepiped spanned by the basis vectors.*

If L has full rank, the determinant of L can be calculated as the absolute value of the determinant of its basis matrix $\mathbf{B}$. Note that the determinant $\det(L)$ is invariant under unimodular basis transformations of $\mathbf{B}$. Equivalently, a lattice L can be defined as a discrete additive subgroup of $\mathbb{Z}^n$.
Let us denote by $||\mathbf{v}||$ the Euclidean ℓ_2-norm of a vector $\mathbf{v}$. Hadamard's inequality [Mey00] relates the length of the basis vectors to the determinant.

Lemma 2.3.2 (Hadamard)

Let $\mathbf{B} = \begin{pmatrix} \mathbf{v_1} \\ \vdots \\ \mathbf{v_n} \end{pmatrix} \in \mathbb{Z}^{n\times n}, n \in \mathbb{N},$ *be an arbitrary non-singular matrix. Then*

$$\det(\mathbf{B}) \leq \prod_{i=1}^{n} ||\mathbf{v_i}|| \,.$$

The successive minima $\lambda_i(L)$ of the lattice L are defined as the minimal radius of a ball containing i linearly independent lattice vectors of L. In a two-dimensional lattice L, basis vectors $\mathbf{b_1}, \mathbf{b_2}$ with lengths $||\mathbf{b_1}|| = \lambda_1(L)$ and $||\mathbf{b_2}|| = \lambda_2(L)$ are efficiently computable via Gaussian reduction.

Theorem 2.3.3
Let $\mathbf{v_1}, \mathbf{v_2} \in \mathbb{Z}^n$ *be basis vectors of a two-dimensional lattice* L*. Then the Gauss-reduced lattice basis vectors* $\mathbf{b_1}, \mathbf{b_2}$ *can be determined in time* $\mathcal{O}(\log^2(\max\{||\mathbf{v_1}||, ||\mathbf{v_2}||\})$*. Furthermore,*

$$||\mathbf{b_1}|| = \lambda_1(L) \text{ and } ||\mathbf{b_2}|| = \lambda_2(L)\,.$$

Information on Gaussian reduction and its running time can be found in [Mey00].
A shortest vector of a lattice satisfies the Minkowski bound, which relates the length of a shortest vector to the determinant and dimension of the lattice.

Lemma 2.3.4 (Minkowski [Min96])
Let L *be an integer lattice with basis matrix* $\mathbf{B} \subseteq \mathbb{Z}^{d\times n}$*. Then* L *contains a non-zero vector* $\mathbf{v}$ *with*

$$||\mathbf{v}|| = \lambda_1(L) \leq \sqrt{d}\det(L)^{\frac{1}{d}}\,.$$

The question remains how to find these small lattice vectors.
In 1982, Arjen K. Lenstra and Henrik W. Lenstra Jr. and László Lovász [LLL82] introduced a way to efficiently determine an approximate shortest vector. They compute a new basis for the lattice in which several conditions are imposed on the vectors.
First, recall the Gram-Schmidt orthogonalization process. Given a set of linearly independent vectors $\mathbf{v_1}, \ldots, \mathbf{v_d}$ the orthogonalized vectors $\mathbf{v_1^*}, \ldots, \mathbf{v_d^*}$ are recursively defined by

$$\mathbf{v_i^*} := \mathbf{v_i} - \sum_{j=1}^{i-1} \mu_{ij}\mathbf{v_j^*} \text{ with } \mu_{ij} = \frac{(\mathbf{v_i}, \mathbf{v_j^*})}{(\mathbf{v_j^*}, \mathbf{v_j^*})}\,.$$

The construction given by Arjen K. Lenstra and Henrik W. Lenstra Jr. and László Lovász modifies this technique. Iteratively sets of Gram-Schmidt orthogonalized vectors are computed and vectors are swapped if they do not fulfill some given conditions concerning their length. This results in a so called LLL-reduced basis.

Theorem 2.3.5 (LLL [LLL82])
Let L be a d-dimensional lattice with basis $\mathbf{v_1}, \ldots, \mathbf{v_d} \in \mathbb{Z}^n$. Then the LLL algorithm outputs a reduced basis $\mathbf{b_1}, \ldots, \mathbf{b_d}$ with the following properties:

- $|\mu_{ij}| \leq \frac{1}{2}$ *for* $1 \leq j < i \leq d$,
- $||\mathbf{b_i^*} + \mu_{i(i-1)}\mathbf{b_{i-1}}^*||^2 \geq \delta ||\mathbf{b_{i-1}^*}||^2$ *for* $1 < i \leq d$.

The running time of this algorithm is $\mathcal{O}(d^4 n(d + \log b_{\max}) \log b_{\max})$, where $b_{\max} \in \mathbb{N}$ denotes the absolute value of the largest entry in the basis matrix. The value δ is chosen in $(\frac{1}{4}, 1]$, in the original work it is $\delta = \frac{3}{4}$.

The running time is the running time of the so-called L^2 algorithm, an efficient LLL version due to Phong Nguyen and Damien Stehlé [NS05].
From the basic properties further conditions on LLL-reduced bases can be derived.

Lemma 2.3.6
Let $\mathbf{b_1}, \ldots, \mathbf{b_d}$ be a basis output by the LLL algorithm. Then it holds that

$$||\mathbf{b_1}|| \leq 2^{\frac{d-1}{4}} \det(L)^{\frac{1}{d}}.$$

For a proof of the theorem and the lemma compare [LLL82].

Another property of an LLL-reduced basis which is especially useful in multivariate settings is given by the following lemma stated by Charanjit S. Jutla in [Jut98]:

Lemma 2.3.7
Let $\mathbf{b_1}, \ldots, \mathbf{b_d}$ be a basis output by the LLL algorithm. Then for $i \leq d$ it holds that

$$||\mathbf{b_i}^*|| \geq 2^{-\frac{i-1}{4}} \left(\frac{\det(L)}{b_{max}^{d-i}} \right)^{\frac{1}{i}}.$$

The value b_{max} denotes the largest absolute value in the basis matrix.

The LLL algorithm only calculates an approximate shortest vector. There are other algorithms with which we can deterministically compute a shortest vector of a lattice L. They work on the Gram-Schmidt orthogonalization of a given lattice basis and enumerate all lattice vectors shorter than some bound $A \geq \lambda(L)$. Such algorithms were introduced by Ravi Kannan [Kan83] and Ulrich Fincke and Michael Pohst [FP83]. However, the running time of these algorithms is exponential. Even given as input an LLL-reduced basis the running time of Fincke and Pohst's algorithm is $2^{\mathcal{O}(d^2)}$. The worst case complexity of Kannan's algorithm is $2^{\frac{d}{2e}+o(d)}$ [HS07]. Here, d denotes the lattice dimension. We state these results ignoring multiplicative factors which are polynomial in the size of the lattice basis. During this thesis, we will use the LLL algorithm as it works well enough for our purposes. This way, we can develop algorithms with polynomial running time.

Small solutions $(\bar{x}_1, \ldots, \bar{x}_l)$ of linear modular multivariate equations

$$f(x_1, \ldots, x_l) := \sum_{i=1}^{l} a_i x_i \equiv 0 \pmod{N}$$

can be determined by shortest vector algorithms. The idea is to embed the given information into a lattice. That is, define a lattice as the set of solutions of $f(x_1, \ldots, x_l) \equiv 0 \pmod{N}$. Then the set $L := \{(x_1, \ldots, x_l) \in \mathbb{Z}^l \mid f(x_1, \ldots, x_l) \equiv 0 \pmod{N}\}$ is indeed a lattice. First, as a subset of $\mathbb{Z}^l$, it is discrete and the addition of elements is associative. Second, as f is linear, it is $f(0, \ldots, 0) \equiv 0 \pmod{N}$, thus, $(0, \ldots, 0) \in L$. Further, if $(x_1, \ldots, x_l) \in L$, then $f(x_1, \ldots, x_l) \equiv 0 \pmod{N} \Leftrightarrow -f(x_1, \ldots, x_l) \equiv 0 \pmod{N} \overset{f \text{ linear}}{\Leftrightarrow} f(-x_1, \ldots, -x_l) \equiv 0 \pmod{N}$. Consequently, $(-x_1, \ldots, -x_l) \in L$. Moreover, L is closed under addition because if $(x_1, \ldots, x_l), (y_1, \ldots, y_l) \in L$, then $f(x_1, \ldots, x_l) \equiv 0 \equiv f(y_1, \ldots, y_l) \pmod{N}$ and, thus, $f(x_1+y_1, \ldots, x_l+y_l) \overset{f \text{ linear}}{\equiv} f(x_1, \ldots, x_l)+f(y_1, \ldots, y_l) \equiv 0 \pmod{N}$. Note that we require f to be a linear polynomial as lattices are linear structures.
If the solution of $f(x_1, \ldots, x_l) \equiv 0 \pmod{N}$ we are searching for is small enough, it corresponds to a shortest vector in the lattice L. Without loss of generality we assume a_l to be invertible modulo N. (Otherwise, we can determine a factor p of N, analyze the two equations $f(x_1, \ldots, x_l) \equiv 0 \pmod{p}$ and $f(x_1, \ldots, x_l) \equiv 0 \pmod{\frac{N}{p}}$, and combine the results.) Let

$$\mathbf{B} = \begin{pmatrix} 1 & & & -a_1 a_l^{-1} \\ & \ddots & & \vdots \\ & & 1 & -a_{l-1} a_l^{-1} \\ & & & N \end{pmatrix}.$$

Then all integer linear combinations of the row vectors of $\mathbf{B}$ which we denote by $\langle \mathbf{B} \rangle$ form the lattice L. We will briefly show this. Let $(x_1, \ldots, x_l) \in L$ and t defined such that $-a_l^{-1} f(x_1, \ldots, x_l) = tN$. Then $(x_1, \ldots, x_l) = (x_1, \ldots, x_{l-1}, -t)\mathbf{B}$. Consequently, $L \subseteq \langle \mathbf{B} \rangle$. Now, let $\mathbf{b}_i = ((\mathbf{b}_i)_1, \ldots, (\mathbf{b}_i)_l)$ denote the i-th row vector of $\mathbf{B}$. Then for $i = 1, \ldots, l-1$ it is $(\mathbf{b}_i)_i = 1, (\mathbf{b}_i)_l = -a_i a_l^{-1}$ and $(\mathbf{b}_i)_j = 0$ for all $j \notin \{i, l\}$. It follows that $f((\mathbf{b}_i)_1, \ldots, (\mathbf{b}_i)_l) \equiv a_i - a_i a_l^{-1} a_l \equiv 0 \pmod{N}$. It is $(\mathbf{b}_l)_l = N$ and $(\mathbf{b}_l)_j = 0$ for $j = 1, \ldots, l-1$. Then $f((\mathbf{b}_l)_1, \ldots, (\mathbf{b}_l)_l) \equiv 0 \pmod{N}$. Therefore, all $\mathbf{b}_i$ belong to L. As f is linear, the same holds for all linear combinations of the basis vectors $\mathbf{b}_i$. This results in $\langle \mathbf{B} \rangle \subseteq L$. Thus, $\langle \mathbf{B} \rangle = L$ and, as the basis vectors $\mathbf{b}_i$ are linearly independent, $\mathbf{B}$ is a basis of L. Then we can determine an approximate shortest vector of L by lattice reduction of $\mathbf{B}$ as described previously.

Let us consider on which conditions this method is going to work. Minkowski's condition (2.3.4) states that the norm of a shortest vector is smaller than or equal to $\sqrt{l} \det(L)^{\frac{1}{l}}$. As $\mathbf{B}$ is an upper triangular matrix we can easily determine $\det(L) = N$. Thus, we can determine $(x_1, \ldots, x_l)$ if its norm is smaller than or equal to $\sqrt{l} N^{\frac{1}{l}}$. Let X_i denote an upper bound on $|\bar{x}_i|$, i.e. $|\bar{x}_i| \le X_i$. We know that $||(\bar{x}_1, \ldots, \bar{x}_l)|| \le ||(X_1, \ldots, X_l)|| \le$

$\sqrt{l}\max\{X_i\}$. This gives the stronger condition $\max\{X_i\}^l \leq N$. For unknowns of equal size this is the same as $\prod_{i=1}^{l} X_i \leq N$. If the sizes of the unknowns are imbalanced, the same result can obtained by adding weights in the lattice basis. For more details consider [May06].

If f is not a modular equation but an integer one, we can proceed analogously. The basis matrix $\mathbf{B}$ is then constructed as before but without the last row which corresponds to the modular reduction. As the basis matrix $\mathbf{B}$ is no longer a square matrix, the calculation of the bound becomes more complicated. An alternative is to choose a large coefficient in the integer equation and regard it as a modular equation modulo exactly this coefficient. Then the analysis of the modular case can be reused.

This technique can be generalized to non-linear equations $f(x_1, \ldots, x_l) = 0$. For the analysis we just insert a linearization step before defining the lattice. Then we can proceed analogously. As an example on how linearization can be used in practice, let us have a look at Michael Wiener's attack from 1990 [Wie90]. Michael Wiener used continued fractions to determine the private RSA exponent d if its absolute value is smaller than one fourth of the public modulus, i. e. $|d| < N^{\frac{1}{4}}$. The same bound can be achieved via linearization [May06, Kat01].

Example 2.3.8
We analyze the RSA key equation $ed \equiv 1 \pmod{\varphi(N)}$ for known values e and N, but unknown factorization of $N = pq$ and unknown d. The factors p and q are of equal bitsize with $p < q$. As we do not know the modulus in this case, we first write the equation as an integer equation

$$ed - k(N - p - q + 1) - 1 = 0\,.$$

Then we take the known value N as new modulus and linearize by setting $x_1 := d$ and $x_2 := k(p + q - 1) - 1$. The resulting equation is

$$f(x_1, x_2) := ex_1 + x_2 \equiv 0 \pmod{N}\,.$$

Let $d < \frac{1}{3}N^{\alpha}$. Then $k(p + q - 1) - 1 < k \cdot 3p < 3N^{\frac{1}{2}+\alpha}$ because $k = \frac{ed-1}{\varphi(N)} < \frac{e}{\varphi(N)}d \overset{e<\varphi(N)}{<} d < N^{\alpha}$, $p < N^{\frac{1}{2}}$ as the smaller factor and p and q of approximately equal size. We would like to determine the solution via the presented technique, namely, by constructing the corresponding lattice L and calculating a shortest vector in it. Thus, there are two conditions on which we can determine the target values. First, they have to correspond to a shortest vector in the lattice L. This is taken as a heuristic assumption. The second condition is necessary for the first one and imposes some size constraints: $\frac{1}{3}N^{\alpha} \cdot 3N^{\frac{1}{2}+\alpha} < N$ which is equivalent to $\alpha < \frac{1}{4}$.
Let $\bar{x}_1$ and $\bar{x}_2$ be the solutions we have determined. Then we directly get the private exponent $d = \bar{x}_1$. To determine the factorization some additional computations are needed. First, k is calculated as $k = \frac{e\bar{x}_1 + \bar{x}_2}{N}$, then $p + q$ is calculated as $p + q = \frac{\bar{x}_2 + 1 + k}{k}$. Then p and q can be determined as roots of $x^2 - (p + q)x + N = 0$.

When linearizing equations, however, a lot of information on the monomials gets lost. In Example 2.3.8 the variable x_1 corresponds to a single variable of the original equation, but x_2 is a function of several variables of total degree two.
As an even worse example of loosing information take the simple equation $f(x) = x^2+x+A \pmod{N}$ for known values A and N. Let $\bar{x}$ denote a root of f. By linearization we derive the equation $\tilde{f}(x) = x_1 + x_2 + Ax_3 \pmod{N}$. From the above analysis we know that all solutions $(\bar{x}_1, \bar{x}_2, \bar{x}_3)$ with $|\bar{x}_1| \leq X_1$, $|\bar{x}_2| \leq X_2$ and $|\bar{x}_3| \leq X_3$ can be determined which fulfill $X_1X_2X_3 < N$. Let $|\bar{x}| \leq N^{\alpha}$. Then we can set $X_1 = N^{2\alpha}$, $X_2 = N^{\alpha}$ and $X_3 = 1$. The bound in this case is $|\bar{x}| \leq N^{\frac{1}{3}}$. However, we are only interested in solutions with $x_1 = x_2^2$. Thus, it is likely that the given bound can be improved if we take the additional knowledge into account.

The basis for these more elaborate analyses is an algorithm which Don Coppersmith presented in 1996 [Cop96b, Cop97]. With the help of this algorithm and some further ideas, Dan Boneh and Glenn Durfee improved Michael Wiener's bound to the value of 0.292 [BD99]. We will not look at their analysis in detail. However, the algorithm introduced by Don Coppersmith will be of use for the analyses we will do in the following chapters.
Therefore, we will present it in what follows. In the algorithm, lattice reduction methods are used to solve multivariate modular and non-modular polynomial equations. The basic idea is to construct sufficiently many polynomials having the same root over the integers. Then a univariate polynomial with the correct solutions over the integers can be determined by the methods presented in Section 2.2. Throughout this thesis we will refer to this method as Coppersmith's method or Coppersmith's algorithm.

The modular univariate variant of Coppersmith's method can be stated as follows:

Theorem 2.3.9 (Coppersmith [Cop97])
Let $f(x)$ be a monic polynomial of degree $\delta \in \mathbb{N}$ in one variable modulo an integer N of unknown factorization. Let X be a bound on the desired solution x_0. If $X < N^{\frac{1}{\delta}}$, then we can find all integers x_0 such that $f(x_0) \equiv 0 \pmod{N}$ and $|x_0| \leq X$ in time $\mathcal{O}(\delta^5(\delta + \log N)\log N)$.

As the method introduced by Don Coppersmith will serve us as a basic tool in many of our analyses, we will describe its major steps here. Generally, given a polynomial $f(x) = \sum_{i=0}^{\delta} a_i x^i$ with $a_i \in \mathbb{Z}$, the algorithm to determine all values $|x_0| \leq X$ such that $f(x_0) = 0 \pmod{N}$ consists of three steps:

1. Building a lattice L such that a solution to the polynomial equation is induced by a reduced basis of a sublattice L_S.

2. Determining an LLL-reduced basis of the sublattice L_S and orthogonalizing it.

3. Determining an equation with the same solutions valid over the integers and solving it.

We will now describe the steps in more detail.

Building a lattice L.
Let $f(x) = \sum_{i=0}^{\delta} a_i x^i$ be the polynomial of which we would like to determine the roots modulo N. Let $\lambda \in \mathbb{N}$, $\lambda \geq 2$. For $i = 0, \dots, \delta - 1$ and $j = 1, \dots, \lambda - 1$ we define the following set of polynomials.

$$f_{ij}(x) = x^i \cdot f(x)^j. \tag{2.9}$$

Throughout the thesis we will call the sets $\mathcal{F}$ of polynomials used to build the lattice **shift polynomials**. The monomials m such that $mf(x)$ is a shift polynomial are denoted by **shift monomials**. The set $\mathrm{Mon}(\mathcal{F})$ is the set of monomials that occur in polynomials of $\mathcal{F}$.
Note that if $f(x_0) \equiv 0 \pmod N$, then $f_{ij}(x_0) \equiv 0 \pmod{N^j}$. With regard to graded lexicographical ordering, the largest monomial occurring in the set of all f_{ij} is $x^{\delta\lambda-1}$. For each polynomial f_{ij} let $\mathbf{f_{ij}}$ be the coefficient vector containing the respective coefficients of the monomials $(1, x, x^2, \dots, x^{\delta\lambda-1})$, i.e. the monomials are ordered from smallest to highest degree. For example, the vector $\mathbf{f_{01}}$ is defined as $(a_0, a_1, \dots, a_\delta, 0, \dots, 0)$.
Let $\mathbf{F_c} := \begin{pmatrix} \mathbf{f_{01}}^T & \dots & \mathbf{f_{(\delta-1)(\lambda-1)}}^T \end{pmatrix}$ and $\mathbf{F_m} := \mathrm{Diag}(N, \dots, N^{\lambda-1})$ be a diagonal matrix with powers of N on the diagonal. The k-th value on the diagonal of $\mathbf{F_m}$ corresponds to the k-th column vector $\mathbf{f_{ij}}^T$ of $\mathbf{F_c}$ and equals N^j. Let $\mathbf{D} = \begin{pmatrix} 1 & & \\ & \dots & \\ & & X^{-\delta\lambda+1} \end{pmatrix}$ be a diagonal matrix with the inverses of the monomials evaluated on the upper bounds on its diagonal.
In practice all values are multiplied by the least common multiple of all denominators so that all calculations can be performed with integer values. This multiplication does not influence the method. For ease of notation in the analysis, however, we work over the rationals here.
Then we define a lattice L via a basis matrix $\mathbf{B}$ as follows:

$$\mathbf{B} := \begin{pmatrix} \mathbf{D} & \mathbf{F_c} \\ \mathbf{0} & \mathbf{F_m} \end{pmatrix}. \tag{2.10}$$

For the desired solution x_0 we have $f(x_0) = y_0 N$ with $y_0 \in \mathbb{Z}$. Let y be a variable denoting this multiple of N. Further, we define the vectors

$$\mathbf{v} = (1, x, x^2, \dots, x^{\delta\lambda-1}, -y, -xy, \dots, -x^{\delta-1}y, -y^2, \dots, -x^{\delta-1}y^{\lambda-1})$$

and

$$\mathbf{v_0} = (1, x_0, x_0^2, \dots, x_0^{\delta\lambda-1}, -y_0, -x_0y_0, \dots, -x_0^{\delta-1}y_0, -y_0^2, \dots, -x_0^{\delta-1}y_0^{\lambda-1}).$$

Then

$$\mathbf{v_0 B} = (1, \frac{x_0}{X}, \frac{x_0^2}{X^2}, \dots, \frac{x_0^{\delta\lambda-1}}{X^{\delta\lambda-1}}, 0, \dots, 0) =: \mathbf{t_0}.$$

This implies that all vectors related to roots of the polynomials f_{ij} are part of the sublattice L_S of vectors with $\delta(\lambda - 1)$ zeros in the end. Thus, our aim is to determine a basis of the sublattice L_S from which we can get an equation which can be solved over the integers.

Determining a suitable basis.
In a first step, we determine a basis of the sublattice. In order to do so, we transform the given basis $\mathbf{B}$ into another basis $\mathbf{B}'$ of the same lattice. The basis $\mathbf{B}'$ should allow to extract a basis of the sublattice L_S easily. Thus, we aim at constructing vectors with zeros in the last components. The other basis vectors restricted to these last components should then form a basis of the lattice corresponding only to the last components. In the best case they also form a basis of $\mathbb{Z}^{\delta(\lambda-1)}$. Such a basis can be transformed to the basis of unit vectors. Thus, the determinant of the lattice only depends on the basis vectors of the sublattice.
As a new basis of the sublattice we can use the basis vectors with zeros in the end shortened to their first components. That is, we would like to transform the original basis to the form

$$\mathbf{B}' = \begin{pmatrix} \mathbf{B_S} & \mathbf{0} \\ * & \mathbf{I}_{\delta(\lambda-\mathbf{1})} \end{pmatrix}, \tag{2.11}$$

where $\mathbf{B_S}$ is a basis matrix of the sublattice we are looking for, and $*$ is some matrix with entries which are not important for our purposes.
A lattice basis can be transformed to another basis of the same lattice by permutation of basis vectors or adding multiples of one basis vector to another basis vector. This is, the transformations we are allowed to make are unimodular transformations. For conditions on which a basis can be transformed accordingly, compare Section 2.1. From now on we only consider the basis $\mathbf{B_S}$ of the sublattice. We perform LLL-reduction on $\mathbf{B_S}$ to get a new LLL-reduced basis $\mathbf{B_R}$.

Calculating the solution.
Having a basis $\mathbf{B_R}$ of the sublattice L_S, we still need a further step in order to get another equation which we can use to determine the solution. Therefore, we quote the following lemma.

Lemma 2.3.10 (Coppersmith [Cop97])

Let L be a lattice and $\mathbf{B} = \begin{pmatrix} \mathbf{b_1} \\ \vdots \\ \mathbf{b_n} \end{pmatrix}$ *be an LLL-reduced basis matrix of L. Further, let*

$D = \det(L)$ be its determinant. Then the following holds:

(a) Any lattice element $\mathbf{t}$ with $||\mathbf{t}|| < D^{\frac{1}{n}} 2^{-\frac{(n-1)}{4}}$ is an element of the hyperplane spanned by $\mathbf{b}_1, \ldots, \mathbf{b_{n-1}}$.

(b) Any lattice element $\mathbf{t}$ with $||\mathbf{t}|| < ||\mathbf{b_j}^*||$ for all $j = k+1, \ldots, n$ is an element of the space spanned by $\mathbf{b}_1, \ldots, \mathbf{b_k}$.

PROOF:

(a) First, note that by combining the conditions of an LLL-reduced basis $\mathbf{B}$ given in Theorem 2.3.5, we get $\left|\left|\mathbf{b_i}^* + \mu_{i(i-1)}\mathbf{b_{i-1}}^*\right|\right|^2 \geq \frac{3}{4}\left|\left|\mathbf{b_{i-1}}^*\right|\right|^2 \Rightarrow \left|\left|\mathbf{b_i}^*\right|\right|^2 + \frac{1}{4}\left|\left|\mathbf{b_{i-1}}^*\right|\right|^2 \geq \frac{3}{4}\left|\left|\mathbf{b_{i-1}}^*\right|\right|^2 \Leftrightarrow 2\left|\left|\mathbf{b_i}^*\right|\right|^2 \geq \left|\left|\mathbf{b_{i-1}}\right|\right|^2$. As $\mathbf{b_1}^*, \ldots, \mathbf{b_n}^*$ form an orthogonal basis, it is $D = \det(\mathbf{B}) = \prod_{i=1}^{n} ||\mathbf{b_i}^*||$. Combining these two results, we obtain

$$\begin{aligned} D \leq \prod_{i=1}^{n} \left(\sqrt{2}\right)^{n-i} ||\mathbf{b_n}^*|| &= 2^{\frac{n(n-1)}{4}} ||\mathbf{b_n}^*||^n \\ \Rightarrow ||\mathbf{b_n}^*|| &\geq D^{\frac{1}{n}} 2^{-\frac{n-1}{4}}\,. \end{aligned} \tag{2.12}$$

Hence, the precondition implies that $||\mathbf{t}|| < ||\mathbf{b_n}^*||$.
Now let us have a closer look at the vector $\mathbf{t}$. As $\mathbf{t} \in L$ we can write $\mathbf{t} = \sum_{i=1}^{n} a_i \mathbf{b_i}$ with $a_i \in \mathbb{Z}$. As the vectors $\mathbf{b_1}, \ldots, \mathbf{b_n}$ and $\mathbf{b_1}^*, \ldots, \mathbf{b_n}^*$ span the same vector space, we can also write $\mathbf{t} = \sum_{i=1}^{n} c_i \mathbf{b_i}^*$ with coefficients $c_i \in \mathbb{R}$. Note, however, that $c_n = a_n$ because $\mathbf{b_n}^* = \mathbf{b_n} - \sum_{i=1}^{n-1} \mu_{ni}\mathbf{b_i}^*$. Consequently, it is $||\mathbf{t}|| \geq |a_n|\,||\mathbf{b_n}^*||$. The inequality is only valid if $a_n = 0$ as $||\mathbf{t}|| < ||\mathbf{b_n}^*||$ as well. This concludes the proof.

(b) From the preconditions we have $||\mathbf{t}|| < ||\mathbf{b_n}^*||$. By the proof of part (a) this implies that $\mathbf{t} = \sum_{i=1}^{n-1} a_i \mathbf{b_i}$. That is, $\mathbf{t}$ is contained in the sublattice spanned by the first $n-1$ basis vectors. Iteratively repeating the argumentation of part (a), we can prove that $a_i = 0$ for all $i = n-1, \ldots, k+1$. This results in the claim. ■

To be able to apply Lemma 2.3.10 we orthogonalize $\mathbf{B_R}$. Let $\mathbf{B_R^*}$ denote the orthogonalized basis.
Now, let $\mathbf{t_S}$ be the vector $\mathbf{t_0}$ reduced to its first $\delta\lambda$ components. Let n correspond to the dimension of the sublattice, namely $n := \delta\lambda$. If the vector is small enough, namely, $||\mathbf{t_0}|| \leq \det(L)^{\frac{1}{n}} 2^{-\frac{(n-1)}{4}}$, then $\mathbf{t_0}$ will be part of the hyperplane spanned by all but the last basis vector $\mathbf{b_n}$. Consequently, $\mathbf{t_0}$ is orthogonal to $\mathbf{b_n}^*$, i. e. $0 = (\mathbf{b_n}^*, \mathbf{t_0}) = \sum_{i=0}^{\delta\lambda-1} (\mathbf{b_n}^*)_{i+1} \frac{x_0^i}{X^i}$. Substituting x_0 by the variable x in this equation and multiplying with $X^{\delta\lambda-1}$, we get $\sum_{i=0}^{\delta\lambda-1} (\mathbf{b_n}^*)_{i+1} x^i X^{\delta\lambda-1-i} = 0$. This is a univariate equation valid over the integers. It can be solved by standard techniques like Newton iteration (e. g. [SB02], Chapter 5) or Sturm sequences ([McN07], Chapter 2).

Conditions on the existence of a solution.
As seen in the previous paragraphs a solution of the modular equation can be determined on condition that the constructed target vector $\mathbf{t_0}$ is small enough. Here we will determine which are the conditions thereof.
By construction we have $||\mathbf{t_0}|| \leq \sqrt{\delta\lambda}$ as $|x_0| \leq X$ and, consequently, $\left(\frac{x_0}{X}\right)^i \leq 1$ for any i. Therefore, applying the above lemma we get the condition

$$\sqrt{\delta\lambda} < \det(L)^{\frac{1}{n}} 2^{-\frac{(n-1)}{4}}\,. \tag{2.13}$$

Calculating $\det(L)$ using the basis $\mathbf{B}$, we derive $\det(L) = X^{-\frac{\delta\lambda(\delta\lambda-1)}{2}} N^{\frac{\delta\lambda(\lambda-1)}{2}}$. Using this and $n = \delta\lambda$, we get the condition

$$\begin{aligned} \sqrt{\delta\lambda} &< X^{-\frac{\delta\lambda-1}{2}} N^{\frac{\lambda-1}{2}} \cdot 2^{-\frac{(\delta\lambda-1)}{4}} \\ \Leftrightarrow X &< N^{\frac{\lambda-1}{\delta\lambda-1}} \cdot 2^{-\frac{1}{2}} (\delta\lambda)^{-\frac{1}{\delta\lambda-1}} . \end{aligned}$$

To see what this condition implies, we can either plug in a specific value for λ or compute the limit with respect to λ. In the first case, we obtain an exact bound and the corresponding lattice to compute it. In the latter case, the bound we obtain is asymptotic. Here the condition asymptotically becomes $X < N^{\frac{1}{\delta}}$. This bound is also obtained by directly requiring

$$\det(L) > 1 . \tag{2.14}$$

In the following, we will refer to this condition as **simplified condition**.
Performing exact calculations, for a given value of $\epsilon > 0$ we can determine a corresponding value $\lambda(\epsilon)$ such that we can calculate all solutions x_0 if $X < N^{\frac{1}{\delta}-\epsilon}$ using a lattice constructed with respect to $\lambda > \lambda(\epsilon)$.
In the case of the example $f(x) = x^2 + x + A \pmod N$, the asymptotic condition is $|\bar{x}| < N^{\frac{1}{2}}$. This is significantly better than the old bound of $|\bar{x}| < N^{\frac{1}{3}}$ we have obtained via linearization.

Coppersmith's method can be generalized to multivariate equations $f(x_1, \ldots, x_l) \equiv 0 \pmod N$ easily. To define the coefficient vectors, an ordering of the occurring monomials has to be defined. Apart from that, decisions have to be made on which polynomials to take as shift polynomials.

Shift polynomials in the multivariate case.
A general strategy on how to choose shift polynomials to build a lattice is described by Ellen Jochemsz and Alexander May in [JM06]. The basic idea of this strategy is to determine all monomials occurring in powers of the original polynomial f. These monomials should form the set of all monomials which occur in the lattice construction. The shift monomials are then defined to ensure this.
We will describe the approach in more detail. The polynomial f is assumed to be monic and to have a non-zero constant term. Depending on a small value $\epsilon > 0$, a positive integer λ is fixed. Then for $k \in \{0, \ldots, \lambda+1\}$ and a value j the sets

$$M_k := \bigcup_{0\le\beta\le t} \{ \prod_{i=1}^{l} x_i^{\alpha_i} x_j^{\beta} \mid \prod_{i=1}^{l} x_i^{\alpha_i} \text{ is a monomial of } f^{\lambda}$$

$$\text{and } \frac{\prod_{i=1}^{l} x_i^{\alpha_i}}{LM(f)^k} \text{ is a monomial of } f^{\lambda-k} \}$$

are defined. The values of t and β are chosen with respect to the specific equation and sizes of the unknowns. The shifts derived from monomials with $\beta > 0$ are called extra

shifts. Such extra shifts can also be applied to several variables.
Then the shift polynomials are defined as

$$f_{\alpha_1,\dots,\alpha_l}(x_1,\dots,x_l) := \frac{\prod_{i=1}^l x_i^{\alpha_i}}{LM(f)^k} f^k(x_1,\dots,x_l) \text{ for } k = 0,\dots,\lambda, \text{ and } \prod_{i=1}^l x_i^{\alpha_i} \in M_k \setminus M_{k+1}\,.$$

Geometrically, this can be interpreted as follows. To any monomial $\prod_{i=1}^l x_i^{\alpha_i}$ we can associate a point $(\alpha_1,\dots,\alpha_l) \in \mathbb{Z}^l$. Applying this correspondence, we can associate a set of points to any polynomial f. We just take the points corresponding to the monomials of f. The convex hull of this set is defined as the **Newton polytope** of f. We denote the Newton polytope by $N(f)$. Then an enlarged and maybe shifted version of the Newton polytope gives rise to the definition of the shift monomials. The shift polynomials are defined such that the set $\mathrm{Mon}(\mathcal{F})$ corresponds to the enlarged and shifted version of $N(f)$.

Having defined the shift polynomial set, we can proceed as in the univariate case. However, it is no longer sufficient to construct only one polynomial $f_n(x_1,\dots,x_l) \in \mathbb{Z}[x_1,\dots,x_l]$ which is valid over the integers. We need to be able to determine the solution efficiently. A sufficient conditions of this is a zero dimensional system of at least l equations. Given such a system, we can determine the solutions by the techniques presented in Section 2.2. Note, however, that the system of equations we obtain is not necessarily of dimension zero. Therefore, the general method is heuristic. Sometimes even fewer equations suffice. This can be seen in the practical experiments in Chapter 6.
The bound obtained this way is

$$\prod_{i=1}^l X_i^{s_i} < N^{s_N}, \text{ for } \begin{cases} s_j = \sum_{\prod_{i=1}^l x_i^{\alpha_i} \in M_0} \alpha_j \\ s_N = \sum_{k=0}^{\lambda} k\left(|M_k| - |M_{k+1}|\right)\,. \end{cases} \tag{2.15}$$

Coppersmith's method over the integers.
In a similar manner Coppersmith's method can be applied to multivariate polynomials over the integers. Roots of univariate integer polynomials can be easily determined using e. g. Newton iteration. However, as soon as we have more than one unknown we need further equations to be able to determine the solutions efficiently. The system formed by the original and the new equations should then be zero dimensional or have other properties that allow to determine its solutions efficiently. In order to get further equations we can again use Coppersmith's algorithm [Cop96a, Cop97]. He described a provable method to solve bivariate equations over the integers.

Theorem 2.3.11 (Coppersmith [Cop97])
Let $f(x_1, x_2)$ be an irreducible polynomial over $\mathbb{Z}$. Further, let X_1, X_2 be bounds on the desired solution $(\bar{x}_1, \bar{x}_2)$. The value W denotes the absolute value of the largest coefficient of $f(x_1X_1, x_2X_2)$.

1. *Assume $f(x_1, x_2)$ to be of degree $\delta \in \mathbb{N}$ in each variable separately. If $X_1X_2 \leq W^{\frac{2}{3\delta}}$, then we can find all integer pairs $(\bar{x}_1, \bar{x}_2)$ such that $f(\bar{x}_1, \bar{x}_2) = 0$ and $|\bar{x}_i| \leq X_i$, $i = 1, 2$, in time polynomial in* $\log W, 2^\delta$.

2. *Suppose $f(x_1, x_2)$ to be of total degree $\delta \in \mathbb{N}$. If $X_1X_2 \leq W^{\frac{1}{\delta}}$, then we can find all integer pairs $(\bar{x}_1, \bar{x}_2)$ such that $f(\bar{x}_1, \bar{x}_2) = 0$ and $|\bar{x}_i| \leq X_i$, $i = 1, 2$, in time polynomial in* $\log W, 2^\delta$.

The proof of this theorem is similar to the proof of Theorem 2.3.9 in the univariate modular case. However, the construction of a suitable basis matrix has to be slightly adopted.

Before building a basis matrix, the set of shift polynomials has to be chosen. In the first case, the set of shift polynomials is defined as $p_{ij}(x_1, x_2) := x_1^i x_2^j f(x_1, x_2)$ with $i, j = 0, \ldots, \lambda$ for a suitable $\lambda \in \mathbb{N}$. In the second case, it is defined as $p_{ij}(x_1, x_2) := x_1^i x_2^j f(x_1, x_2)$ such that $i + j \leq \lambda$ for a suitable $\lambda \in \mathbb{N}$.
Note that in the modular case the right side of the basis matrix $\mathbf{B}$ given in (2.10) consists of two parts, one part corresponding to the polynomials by containing their coefficients, the second part corresponding to the moduli. In the integer case, however, there are no moduli. Thus, we have to build the matrix using only the coefficient vectors of the polynomials.
We present here one method to build a basis matrix in the integer case similar to the one in [BM05]. This is basically the method we will use for analyses in more complex cases in the following chapters as well.
Let $\mathcal{F}$ denote the set of all shift polynomials and $\text{Mon}(\mathcal{F})$ the set of all monomials occurring in $\mathcal{F}$. Let m be the monomial corresponding to the largest coefficient of $f(x_1X_1, x_2X_2)$, $|\mathcal{F}|$ denote the number of shift monomials and $m_1, \ldots, m_{|\mathcal{F}|}$ be an ordering of all shift monomials such that the monomial $m_i m$ does not occur in the set of monomials of $m_j f(x_1, x_2)$ for any $j < i$. Let $w := |\text{Mon}(\mathcal{F})|$ be the number of monomials occurring in $\mathcal{F}$ and let $m_{|\mathcal{F}|+1}, \ldots, m_w$ be an ordering of the monomials of $\text{Mon}(\mathcal{F}) \setminus \{m_1 m, \ldots, m_{|\mathcal{F}|} m\}$. Furthermore, let M_i denote the evaluation of m_i in the values (X_1, X_2) with $i = |\mathcal{F}| + 1, \ldots, w$. We set $\mathbf{x} := (m_{|\mathcal{F}|+1}, \ldots, m_w, m_1 m, \ldots, m_{|\mathcal{F}|} m)$. Let $\mathbf{D} := \text{Diag}(M_{|\mathcal{F}|+1}^{-1}, \ldots, M_w^{-1})$. For any polynomial $p_{ij}(x_1, x_2)$ let $\mathbf{p_{ij}}$ denote the coefficient vector of the monomials in p_{ij} ordered such that $\mathbf{p_{ij}} \mathbf{x}^T = p_{ij}(x_1, x_2)$. Let $\mathbf{F}$ be the matrix consisting of the coefficient vectors $(\mathbf{p_{ij}})^T$ for all $p_{ij} \in \mathcal{F}$. Using these definitions, we define the lattice L via a basis matrix

$$\mathbf{B} := \begin{array}{c} \left(\begin{array}{cc} \mathbf{D} & \\ & \mathbf{F} \\ \mathbf{0} & \end{array} \right) \end{array} \begin{array}{c} m_{|\mathcal{F}|+1} \\ \vdots \\ m_w \\ m_1 m \\ \vdots \\ m_{|\mathcal{F}|} m \end{array} .$$

Like in the modular case the basis matrix $\mathbf{B}$ is an upper triangular matrix. The diagonal values due to $\mathbf{F}$, however, are no longer derived from any moduli but correspond to a

(preferably) large coefficient of a shift polynomial. Furthermore, we can no longer take advantage of powers of the initial polynomial f, but only multiply f with monomials to obtain shift polynomials.
From this point on we can proceed in the same manner as in the modular case. We perform lattice reduction on $\mathbf{B}$. If the size conditions are fulfilled, we provably get a second equation $g(x_1, x_2)$ in the two unknowns such that f and g form a system of dimension zero. Then techniques like Groebner basis computations or calculation of a resultant presented in Section 2.2 can be used to determine the solution.

In [BM05], Johannes Blömer and Alexander May generalize this approach. They give further constructions of shift polynomial sets and show that solving a modular univariate polynomial equation with a composite modulus of unknown factorization can be reduced to solving a bivariate integer equation. The shift polynomial sets they use, however, are required to have a certain property. This property ensures that the newly determined polynomial is coprime to the initial one.
In contrast to their construction, the monomial sets $\{m_1 m, \dots, m_{|\mathcal{F}|} m\}$ and $\mathrm{Mon}(\mathcal{F})$ we use may have any structure. This allows to compute better bounds. On the negative side, however, it might happen that a newly constructed polynomial is a multiple of the first one or that both polynomials have a non-trivial common divisor. Then we do not get any new information from the new polynomial. This implies that the solution cannot be recovered efficiently. However, in practice, the method usually works even without being provable so that this can be used as a heuristic. In more general cases of more variables or systems of equations a heuristic is usually needed anyway. Hence, in our analyses, we would have to include a heuristic even in the bivariate case. We will use this technique to analyze the problem of implicit factoring in Chapter 6. For the description of this problem, however, we need at least three variables.

Other variants to build a basis matrix and their analyses can be found in [Cop97, Cor04, Cor07].

An application of this method to the problem of factorization with partially known factors is given by Don Coppersmith.

Theorem 2.3.12 ([Cop97] Theorem 5)
Let N be an n-bit composite number. Then we can find the factorization of $N = pq$ in polynomial time if we know the low order $\frac{n}{4}$ bits of p.

Like in the modular case Coppersmith's method can be generalized to multivariate equations over the integers in $k > 2$ unknowns as well. However, to solve equations over the integers in more than $k > 2$ variables we need to determine more than one additional equation. One sufficient condition to be able to determine the roots is that we can determine $k - 1$ new equations such that they, together with the original equation, form a system of dimension zero. Then we can determine the solutions by successive resultant computation or the use of Groebner bases described in Section 2.2.

In his work, Don Coppersmith includes the condition that suitable equations are generated as a heuristic.

In 2007, Aurélie Bauer and Antoine Joux [BJ07] showed how this condition can be provably achieved in the case of three variables. The drawbacks of their method, however, are an increased running time and smaller bounds on the size of the solutions which can be determined. A short description of their method is given in Chapter 6.
Due to the worse bounds and the fact that the heuristic approach usually works well in practice, we will stick to the heuristic methods in the subsequent chapters.
Some examples of different attacks with Coppersmith's method for which the heuristic is practically verified are given in [BD99, JM06, JM07, HM08].
However, different heuristics may be used as well. For example, Santanu Sarkar and Subhamoy Maitra do not get a system of dimension zero [SM09] when analyzing the problem of implicit factoring of two integers. Nevertheless, a Groebner basis of the set of polynomial equations reveals the solution. Additional information on this will be given in Section 6.1.

Chapter 3

Solving Systems of Modular Univariate Polynomial Equations

In this chapter we deal with the problem of solving univariate equations or systems of univariate equations. Assume $f(x)$ is an arbitrary univariate polynomial. Our aim is to determine all values x_0 such that $f(x_0) = 0$. If $f(x) \in \mathbb{Z}[x]$ is a polynomial over the integers (or rationals or reals), its roots can efficiently be found by Newton iteration [SB02] or binary search.

Let us now consider modular polynomials $f(x) \in \mathbb{Z}_N[x]$, $N \in \mathbb{N}$. If N is prime or a prime power, then again the equation $f(x) \equiv 0 \pmod{N}$ can be solved using e. g. Berlekamp's algorithm [Ber67, Ber70, BS96]. In its basic form, however, the algorithm has a complexity exponential in n, the number of factors of $f(x)$. Mark van Hoeij improved the complexity to a polynomial time complexity in n by using lattice reduction [vH01]. The polynomial complexity bound for this algorithm is given in [BvHKS07]. Note that n is not necessarily polynomial in the bitsize of N. In the sequel of this chapter we will without loss of generality assume all given moduli to be composite.

For arbitrary moduli the best result known so far was given by Don Coppersmith (Theorem 2.3.9). We briefly recall the result here. Let $\delta := \deg(f)$ be the degree of f. Then all $|x_0| < N^{\frac{1}{\delta}}$ such that $f(x_0) \equiv 0 \pmod{N}$ can be determined in polynomial time in $\log(N)$ and δ. In [Cop01], Don Coppersmith further argues that by this method the bound cannot be improved for general polynomials. Let $N = q^3$ for a prime q and set $f(x) := x^3 + Dqx^2 + Eq^2x$ with $D, E \in \mathbb{Z}$. Any x_0 which is a multiple of q clearly solves $f(x_0) \equiv 0 \pmod{N}$. For $\epsilon > 0$ the number of x_0 such that $|x_0| < N^{\frac{1}{3}+\epsilon} = qN^{\epsilon}$ and $f(x_0) \equiv 0 \pmod{N}$ is about $2N^{\epsilon}$. That is, we have exponentially many such values x_0 and cannot hope to find them with Coppersmith's algorithm as the number of small roots is bounded by the lattice dimension. Furthermore, the running time of the algorithm is polynomial in the lattice dimension as well. That is, we cannot even output exponentially many solutions in polynomial time.

Sometimes, however, we get additional equations with the same solution. The question arises what happens then. Does this information help to determine larger solutions as well? Instead of one equation, let us now consider the following system of $k \in \mathbb{N}$ equations with a shared solution x_0. We define the problem of solving *systems of modular univariate polynomial equations* (SMUPE-problem).

Definition 3.0.13 (SMUPE-problem)
Let $k \in \mathbb{N}$, $\delta_1, \ldots, \delta_k \in \mathbb{N}$, and $N_1, \ldots, N_k \in \mathbb{N}$. Suppose $N_1 \leq N_2 \leq \ldots \leq N_k$. Assume $f_1(x), \ldots, f_k(x)$ to be polynomials of degree $\delta_1, \ldots, \delta_k$ in $\mathbb{Z}_{N_1}[x], \ldots, \mathbb{Z}_{N_k}[x]$, respectively. Let

$$\begin{array}{rcl} f_1(x) & \equiv & 0 \pmod{N_1} \\ f_2(x) & \equiv & 0 \pmod{N_2} \\ & \vdots & \\ f_k(x) & \equiv & 0 \pmod{N_k} \end{array} \tag{3.1}$$

be a system of univariate polynomial equations.

Let $X < N_1$, $X \in \mathbb{R}$. Find all common roots x_0 of (3.1) with size $|x_0| \leq X$.

We would like to analyze up to which size of the unknown the solutions can be found in polynomial time. This depends on the structure of the given set of equations. Without loss of generality we assume that for all N_i and N_j, $i, j = 1, \ldots, k$, it is either $N_i = N_j$, or N_i and N_j are relatively prime. In any other case we can determine factors of N_i and N_j by calculating their greatest common divisor $\gcd(N_i, N_j) = N_{ij}$. Then we can transform the given equation into equations modulo these factors by the Chinese Remainder Theorem (Theorem 2.2.13). More precisely, the equation $f_i(x) \equiv 0 \pmod{N_i}$ is equivalent to a system $f_{i1}(x) \equiv 0 \pmod{N_{ij}}$ and $f_{i2}(x) \equiv 0 \pmod{\frac{N_i}{N_{ij}}}$. Analogously, $f_j(x) \equiv 0 \pmod{N_j}$ is equivalent to a system $f_{j1}(x) \equiv 0 \pmod{N_{ij}}$ and $f_{j2}(x) \equiv 0 \pmod{\frac{N_j}{N_{ij}}}$. Putting the equations $f_{i1}(x) \equiv 0 \pmod{N_{ij}}, f_{i2}(x) \equiv 0 \pmod{\frac{N_i}{N_{ij}}}, f_{j1}(x) \equiv 0 \pmod{N_{ij}}, f_{j2}(x) \equiv 0 \pmod{\frac{N_j}{N_{ij}}}$ into the system instead of $f_i(x) \equiv 0 \pmod{N_i}$ and $f_j(x) \equiv 0 \pmod{N_j}$, we obtain a system with equal or coprime moduli. If this system contains a prime modulus p, we can determine x_0 by regarding only the equation with this modulus. Therefore, we still assume all moduli to be composite.
To better analyze the system, we distinguish the following two cases: The subsequent section deals with systems of equations of which at least two share the same modulus. The case of mutually coprime moduli is analyzed in Section 3.2.

3.1 Solving Systems of Modular Univariate Polynomial Equations with a Common Modulus (SMUPE1)

Often two equations $f_1(x) \equiv 0 \pmod{N}$ and $f_2(x) \equiv 0 \pmod{N}$ are sufficient to determine all common solutions. Let us start with two examples where this is easy to see as the given systems of two equations have a special structure. Both examples are derived from the context of RSA encryption. An unknown message m is encrypted by exponentiation by a public exponent e to a known ciphertext $c \equiv m^e \pmod{N}$. Determining m corresponds to finding the solutions of the equation $f(x) := x^e - c \equiv 0 \pmod{N}$. Combining more than one of these equations with the same root can lead to easy algorithms to calculate m.

Example 3.1.1 (G. Simmons [Sim83])
Let $e_1, e_2 \in \mathbb{N}$ with $\gcd(e_1, e_2) = 1$ and $N \in \mathbb{N}$ be composite. Let $f_1(x) := x^{e_1} - m^{e_1}$ and $f_2(x) := x^{e_2} - m^{e_2}$ be two polynomials in $\mathbb{Z}_N[x]$. Our aim is to find the common root m of $f_1(x)$ and $f_2(x)$. To achieve this, we compute integers u_1, u_2 such that $u_1e_1 + u_2e_2 = 1$ with the help of the Extended Euclidean Algorithm. This gives us $m \equiv (m^{e_1})^{u_1}(m^{e_2})^{u_2} \pmod{N}$. The running time of this attack is polynomial in the bitlength of (e_1, e_2) as the Extended Euclidean Algorithm and exponentiation are.

In practice such equations occur in a plain RSA scenario in which the same message m is sent to two users with coprime public exponents e_1 and e_2 and common public moduli $N_1 = N_2 = N$.

In a different scenario, instead of sending the same message twice to different users, similar messages can be sent to the same user.
At the Crypto'95 rump session Matthew K. Franklin and Michael K. Reiter presented an algorithm to recover linearly related messages, both encrypted with the RSA public key $e = 3$ and N.

Example 3.1.2 (Franklin, Reiter [FR95])
Let m_1 and $m_2 = m_1 + 1$ be two unknown messages. And let $c_1 \equiv m_1^3 \pmod{N}$ and $c_2 \equiv m_2^3 \pmod{N}$ be their known RSA encryptions under the RSA public key $(N, 3)$. Then the messages can be recovered by calculating

$$\begin{aligned} \frac{c_2 + 2c_1 - 1}{c_2 - c_1 + 2} &\equiv \frac{(m_1+1)^3 + 2m_1^3 - 1}{(m_1+1)^3 - m_1^3 + 2} \\ \equiv \frac{3m_1^3 + 3m_1^2 + 3m_1}{3m_1^2 + 3m_1 + 3} &\equiv m_1 \pmod{N} \\ \textit{and } m_2 &\equiv m_1 + 1 \pmod{N}. \end{aligned}$$

For any other affinely related messages m_1 and $m_2 = \alpha m_1 + \beta$, $\alpha \in \mathbb{Z}_N^*, \beta \in \mathbb{Z}_N$ encrypted under the RSA public key $(N, 3)$, the messages can be recovered in a similar way:

$$\begin{aligned} \frac{\beta(c_2 + 2\alpha^3 c_1 - \beta^3)}{\alpha(c_2 - \alpha^3 c_1 + 2\beta^3)} &\equiv \frac{\beta((\alpha m_1 + \beta)^3 + 2\alpha^3 m_1^3 - \beta^3)}{\alpha((\alpha m_1 + \beta)^3 - \alpha^3 m_1^3 + 2\beta^3)} \\ \equiv \frac{3\alpha^3\beta m_1^3 + 3\alpha^2\beta^2 m_1^2 + 3\alpha\beta^3 m_1}{3\alpha^3\beta m_1^2 + 3\alpha^2\beta^2 m_1 + 3\alpha\beta^3} &\equiv m_1 \pmod{N} \\ \text{and } m_2 &\equiv \alpha m_1 + \beta \pmod{N}. \end{aligned}$$

This approach was generalized to other public exponents, more equations and relations of different degrees by Don Coppersmith, Matthew K. Franklin, Jacques Patarin and Michael K. Reiter in [CFPR96].

In a first step, they relax the restriction on e and regard any two equations $m_1^e \equiv c_1 \pmod{N}$ and $m_2^e \equiv c_2 \pmod{N}$. Following the approach described in Example 3.1.2, one can construct two equations $P(m_1)$ and $Q(m_1)$ such that $Q(m_1) \equiv m_1 P(m_1) \pmod{N}$ using only the publicly known parameters. This computation, however, is quite complicated.

For larger exponents e, Don Coppersmith et al. pursue a different approach. Let again $m_1^e \equiv c_1 \pmod{N}$ and $m_2^e \equiv c_2 \pmod{N}$ be encryptions of two unknown messages m_1 and m_2 related by a polynomial $p(x)$ of degree $\deg(p) = \delta$ such that $m_2 = p(m_1)$. We transform these equations to $m_1^e - c_1 \equiv 0 \pmod{N}$ and $p^e(m_1) - c_2 \equiv 0 \pmod{N}$. Then m_1 is a common solution of the two equations $f_1(x) := x^e - c_1 \equiv 0 \pmod{N}$ and $f_2(x) := p^e(x) - c_2 \equiv 0 \pmod{N}$. Consequently, $f_1(x)$ and $f_2(x)$ share a factor $(x - m_1)$. Computing the greatest common divisor $\gcd(f_1(x), f_2(x)) \pmod{N}$ reveals this factor if it is the only common factor and the computation does not fail. We will comment on both problems in the subsequent paragraph with respect to more general equations. The running time of this method is $\mathcal{O}(e\delta \log^2(e\delta) \log^2(N))$.

Further extending the problem, one might have an implicit relation $p(m_1, m_2) = 0$ between m_1 and m_2 instead of the explicit one $m_2 = p(m_1)$. The given equations are then multivariate and the analysis becomes more complicated. We will refer to this in Section 4.1.

Another interesting generalization will be the topic of the rest of this section. It is a generalization from RSA polynomials to arbitrary univariate polynomials. Let $f_1(x)$ and $f_2(x) \in \mathbb{Z}_N[x]$ be polynomials of degree δ_1 and δ_2, respectively. The goal is again to find all solutions x_0 such that $f_1(x_0) \equiv 0 \equiv f_2(x_0) \pmod{N}$. This system can be solved in the same manner as the RSA equations given before. Just compute $\gcd(f_1(x), f_2(x)) \pmod{N}$. If both polynomials share exactly one common root (with multiplicity one), a linear factor $f(x) := \alpha x + \beta$ is revealed. Thus, $x_0 \equiv -\beta\alpha^{-1} \pmod{N}$.
However, two problems may occur during the computation of a greatest common divisor. First, the computation might be impossible. This happens whenever the leading coefficient c of a polynomial which is computed by the Euclidean Algorithm is not invertible in $\mathbb{Z}_N$. Then $\gcd(c, N) > 1$, and we have found a divisor of N. Thus, we can split up the equations modulo N into two equations modulo $\gcd(c, N)$ and $\frac{N}{\gcd(c,N)}$ by the Chinese Remainder Theorem. If the new moduli are prime, we can determine x_0 easily. Otherwise we can restart the algorithm with smaller moduli. In the following we, therefore, assume that the greatest common divisor computation always succeeds.
Even if successful, the result of the greatest common divisor computation might be a polynomial g of degree δ_g greater than 1. In this case we cannot compute all possible solutions of the system under consideration but only all solutions x_0 such that $|x_0| < N^{\frac{1}{\delta_g}}$. These solutions x_0 can be determined by applying Coppersmith's method to the greatest common divisor polynomial g. Then further equations with the same solution might help to reveal a factor of smaller degree.

3.2 Solving Systems of Modular Univariate Polynomial Equations with Coprime Moduli (SMUPE2)

In Section 3.1 we have seen that given a system of modular equations a common solution can usually be determined from any two equations with equal moduli. Therefore, we now assume that no pair of such equations occurs in our system, i. e. we focus on the analysis of univariate systems of equations with pairwise coprime moduli. Let us formally state the problem of solving *systems of modular univariate polynomial equations with mutually coprime moduli* (SMUPE2-problem). It is a special case of Definition 3.0.13.

Definition 3.2.1 (SMUPE2-problem)
Let $k \in \mathbb{N}$, $\delta_1, \ldots, \delta_k \in \mathbb{N}$, and let $N_1, \ldots, N_k \in \mathbb{N}$ be mutually coprime composite numbers of unknown factorization. Suppose $N_1 < N_2 < \ldots < N_k$. Assume $f_1(x), \ldots, f_k(x)$ to be polynomials of degree $\delta_1, \ldots, \delta_k$ in $\mathbb{Z}_{N_1}[x], \ldots, \mathbb{Z}_{N_k}[x]$, respectively. Let

$$\begin{array}{rcll} f_1(x) & \equiv & 0 & \pmod{N_1} \\ f_2(x) & \equiv & 0 & \pmod{N_2} \\ & \vdots & & \\ f_k(x) & \equiv & 0 & \pmod{N_k} \end{array} \tag{3.2}$$

be a system of univariate polynomial equations.

Let $X < N_1$, $X \in \mathbb{R}$. Find all common roots x_0 of (3.2) with size $|x_0| \leq X$.

Solving SMUPE2 is, thus, equivalent to determining all common solutions up to a certain bound of a given system of modular univariate equations.
Johan Håstad [Hås88] gave the following algorithm for solving the SMUPE2-problem. Let $\delta \in \mathbb{N}$ be the maximum degree of all polynomials occurring in the system, i. e. $\delta := \max_{i=1,\ldots,k}\{\delta_i\}$. One first multiplies the given polynomials with $x^{\delta-\delta_i}$ to adjust their degrees. Then one combines the resulting polynomials using the Chinese Remainder Theorem to a univariate polynomial $f(x)$ with the same roots modulo $\prod_{i=1}^{k} N_i$. Applying lattice reduction methods, Johan Håstad derived $k > \frac{\delta(\delta+1)}{2}$ as a lower bound on the number of polynomials for efficiently finding all roots x_0 with $|x_0| < N_1$. As $f(x)$ is a polynomial of degree δ, this bound can be easily improved to $k \geq \delta$ by directly applying Coppersmith's lattice-based techniques [Cop97] to $f(x)$ (see e.g. [Bon99]).

An Approach with Better Polynomial Modeling

We give a different construction to combine all k polynomial equations into a single equation $f(x) \equiv 0 \pmod{\prod_{i=1}^{k} N_i}$ in [MR08]. Instead of multiplying the polynomials by powers of x like in Håstad's approach, we take powers of the polynomials $f_i(x)$ themselves. This results in the condition $\sum_{i=1}^{k} \frac{1}{\delta_i} \geq 1$ for solving the SMUPE2-problem for all x_0 with $|x_0| < N_1$. In case all polynomials share the same degree δ, this corresponds to the condition $k \geq \delta$.

For polynomials of different degrees, however, our new condition is superior. In particular, a few polynomials of low degree suffice to calculate all joint solutions.

As an introductory example let us consider Coppersmith's method (Theorem 2.3.9) for the first equation $f_1(x) \equiv 0 \pmod{N_1}$ in (3.2). This way, only small roots x_0 with $|x_0| < N_1^{\frac{1}{\delta_1}}$ can be found in polynomial time. By regarding further equations, this bound can be improved until all solutions can be found eventually.
By Håstad's algorithm in combination with Theorem 2.3.9 the condition $k \geq \delta$ with $\delta := \max_{i=1,\ldots,k}\{\delta_i\}$ is sufficient to solve a system of equations efficiently. However, this condition is clearly not optimal as the following trivial example shows. Let $N_1 < \ldots < N_4$ and take the following equations:

$$\begin{aligned} x^3 &\equiv c_1 \pmod{N_1} \\ x^3 &\equiv c_2 \pmod{N_2} \\ x^3 &\equiv c_3 \pmod{N_3} \\ x^5 &\equiv c_4 \pmod{N_4}. \end{aligned}$$

Then $k = 4 < 5 = \delta$, i.e. Håstad's condition is not fulfilled. However, if we just take the first three equations, we are able to compute all common solutions smaller than N_1. This indicates that we should take the proportion of higher and lower degrees of the polynomials into account. Let us now change the given example a little bit into a non-trivial one so that no subsystem of the equations fulfills the sufficient condition:

$$\begin{aligned} x^3 &\equiv c_1 \pmod{N_1} \\ x^3 &\equiv c_2 \pmod{N_2} \\ x^5 &\equiv c_3 \pmod{N_3} \\ x^5 &\equiv c_4 \pmod{N_4}. \end{aligned}$$

The parameters k and δ as well as the N_i remain the same. Can we still determine all solutions? We notice that we can transform the first equation by squaring into

$$x^6 \equiv 2c_1x^3 - c_1^2 \pmod{N_1^2}.$$

Applying Theorem 2.3.9 to this equation, we can find all solutions x_0 for which $|x_0| < (N_1^2)^{\frac{1}{6}} = N_1^{\frac{1}{3}}$ holds. This is the same bound which we get for the solutions of the original equation $x^3 \equiv c_1 \pmod{N_1}$. We proceed with the second equation in the same way, then multiply the two other equations by x and finally combine all the equations by the Chinese Remainder Theorem (Theorem 2.2.13). By this we obtain

$$x^6 \equiv a_1(2c_1x^3 - c_1^2) + a_2(2c_2x^3 - c_2^2) + a_3xc_3 + a_4xc_4 \pmod{N_1^2N_2^2N_3N_4},$$

where the a_i are the coefficients from the Chinese Remainder Theorem, i.e. $a_i \equiv 1 \pmod{N_i}$, $a_i \equiv 0 \pmod{N_j}$, $j \neq i$. The above equation can be solved applying Coppersmith's algorithm for x_0 with $|x_0| < (N_1^2N_2^2N_3N_4)^{\frac{1}{6}}$. This condition is fulfilled for any

x_0 with $|x_0| < N_1 = (N_1^6)^{\frac{1}{6}} \le (N_1^2 N_2^2 N_3 N_4)^{\frac{1}{6}}$. Therefore, we can determine all solutions of the above system of equations, although the condition $k \ge \delta$ is not fulfilled.
In order to generalize our approach, we make the following crucial observation. Let $f(x)$ be a polynomial of degree δ. Let $f(x) \equiv 0 \pmod{N}$ for $N \in \mathbb{N}$, and let $m \in \mathbb{N}$. Then $g(x) := f^m(x) \equiv 0 \pmod{N^m}$. The solutions x_0 with $|x_0| < N$ of the two equations remain unchanged. Moreover, with Coppersmith's Theorem 2.3.9 we can determine those solutions x_0 for which the condition $|x_0| < N^{\frac{1}{\delta}} \Leftrightarrow |x_0| < (N^m)^{\frac{1}{m\delta}}$ holds. Thus, Coppersmith's bound is invariant under taking powers of the polynomial $f(x)$.
As opposed to our approach, in Håstad's algorithm one does not take powers of the polynomials but multiplications of polynomials with powers of x. This increases the degree of the polynomial but leaves the modulus unchanged. Let $f(x)$ be a polynomial of degree δ with $f(x) \equiv 0 \pmod{N}$ for $N \in \mathbb{N}$. Then with $\gamma > \delta$ the equation $g(x) := x^{\gamma-\delta} f(x) \equiv 0 \pmod{N}$ contains all the solutions x_0 of $f(x) \equiv 0 \pmod{N}$ with $|x_0| < N$. However, applying Coppersmith's method to determine roots of $g(x)$ we only get roots x_0 with $|x_0| < N^{\frac{1}{\gamma}} < N^{\frac{1}{\delta}}$. So obviously, Coppersmith's bound is not invariant under multiplication with powers of x. This explains why we obtain a superior bound on the size of the roots.
In the following analysis we will restrict ourselves to monic polynomials. If one of the given polynomials $f_i(x)$ is not monic, either the coefficient of the leading monomial is invertible, or we can find a factor of the modulus. In the first case, we transform the polynomial to a monic one by multiplication with the inverse of the leading coefficient. In the latter case, we can replace the modular equation $f_i(x) \equiv 0 \pmod{N}$ by two equations modulo the two factors. For RSA moduli we even obtain the complete factorization, which in turn allows for efficiently solving this polynomial equation modulo the prime factors provided that the degree δ_i is polynomial in $\log(N)$.

Theorem 3.2.2
Let (f_i, δ_i, N_i), $i = 1, \ldots, k$, be an instance of the SMUPE2-problem with monic f_i. Define $M := \prod_{i=1}^k N_i^{\frac{\delta}{\delta_i}}$ with $\delta := lcm\{\delta_i, i = 1, \ldots, k\}$. Then the SMUPE2-problem can be solved for all x_0 with

$$|x_0| < M^{\frac{1}{\delta}}$$

in time $\mathcal{O}(\delta^6 \log^2 M)$.

Proof: Let x_0 be a solution of the system of polynomial equations (3.2). Then x_0 is a solution of

$$f_i^{\frac{\delta}{\delta_i}}(x) \equiv 0 \pmod{N_i^{\frac{\delta}{\delta_i}}} \text{ for all } i = 1, \ldots, k\,.$$

All these equations have common degree δ and are monic. Combining them by Chinese Remaindering yields a polynomial $f(x)$ of degree δ such that x_0 is a solution of $f(x) \equiv 0 \pmod{M}$ with $M := \prod_{i=1}^k N_i^{\frac{\delta}{\delta_i}}$. Moreover, this polynomial is still monic. For the coefficient a_δ of the monomial x^δ in $f(x)$ it holds that $a_\delta \equiv 1 \pmod{N_i^{\frac{\delta}{\delta_i}}}$ for all $i = 1, \ldots, k$ and, therefore, $a_\delta \equiv 1 \pmod{M}$.

The above step can be performed in time $\mathcal{O}(\delta \log^2 M)$ by Theorem 2.2.13. With Theorem 2.3.9 all solutions x_0 of the above equation which fulfill $|x_0| \leq M^{\frac{1}{\delta}} = (\prod_{i=1}^{k} N_i^{\frac{\delta}{\delta_i}})^{\frac{1}{\delta}}$ can be found in time $\mathcal{O}(\delta^5(\delta + \log M) \log M)$. Therefore, the result can be obtained in time $\mathcal{O}(\delta^6 \log^2 M)$. ■

Theorem 3.2.2 immediately gives us a sufficient condition on k and the δ_i for solving the SMUPE2-problem for all $x_0 \in \mathbb{Z}_{N_1}$.

Corollary 3.2.3
The SMUPE2-problem can be solved for all $x_0 \in \mathbb{Z}_{N_1}$ in time $\mathcal{O}(\delta^6 \log^2 M)$ provided that

$$\sum_{i=1}^{k} \frac{1}{\delta_i} \geq 1. \tag{3.3}$$

PROOF: Let x_0 be a common solution to all the equations. An application of Theorem 3.2.2 gives us $|x_0| < M^{\frac{1}{\delta}} := (\prod_{i=1}^{k} N_i^{\frac{\delta}{\delta_i}})^{\frac{1}{\delta}}$ as an upper bound for all roots that can be computed in time $\mathcal{O}(\delta^6 \log^2 M)$. As $(\prod_{i=1}^{k} N_i^{\frac{\delta}{\delta_i}})^{\frac{1}{\delta}} \geq \prod_{i=1}^{k} N_1^{\frac{1}{\delta_i}} = N_1^{\sum_{i=1}^{k} \frac{1}{\delta_i}} \geq N_1$ all solutions $x_0 \in \mathbb{Z}_{N_1}$ can be found. ■

By this we get an algorithm to solve the SMUPE2-problem with running time polynomial in the bitsize of the N_i, $i = 1, \ldots, k$, if δ is polynomial in the bitsize of the N_i.

Remark 3.2.4
The same result is obtained by applying Coppersmith's method [Cop97] directly to the polynomials $f_1(x), \ldots, f_k(x)$ instead of $f(x)$. The polynomial modeling is then revealed in the choice of shift polynomials. To do so, however, we need some further information on applying Coppersmith's algorithm to systems of equations. Therefore, the corresponding proof will be given in Section 5.3.1.

Comparing this to the result due to Håstad and Coppersmith, we observe that in the case $\delta := \delta_1 = \ldots = \delta_k$ the sufficient condition is $k \geq \delta$ with both methods. For different δ_i however, our method is always superior. Taking e.g. the illustrating example with public exponents $(3, 3, 5, 5)$ from the beginning of this section, we see that our new condition $\frac{1}{3} + \frac{1}{3} + \frac{1}{5} + \frac{1}{5} = \frac{16}{15} \geq 1$ is fulfilled.
Moreover, our formula captures the intuition that equations of low degree δ_i comprise more information since they contribute to the sum in (3.3) with a larger term $\frac{1}{\delta_i}$ than equations with higher degree.

3.2.1 Optimality of Our Bound for Solving SMUPE2

In this section, we will see that the condition $|x_0| < M^{\frac{1}{\delta}}$ for efficiently solving the SMUPE2-problem given in Theorem 3.2.2 is optimal if the moduli N_i are prime powers. This implies that the condition cannot be improved in general, unless we make use of the structure of

the moduli or of the specific polynomials occurring in the system. Thus, our argument does not exclude the existence of superior conditions for special moduli, e.g. square-free N_i.
The counting argument that we use is a generalization of the argument in [Cop01] to systems of polynomial equations instead of a single equation.
Let $k \in \mathbb{N}$. Let $p_1, \ldots, p_k$ be different prime numbers and $\delta_1, \ldots, \delta_k \in \mathbb{N}$. We define $N_1 := p_1^{\delta_1}, \ldots, N_k := p_k^{\delta_k}$. Suppose $N_1 < \ldots < N_k$. Let us look at the following system of polynomial equations:

$$\begin{aligned} f_1(x) := x^{\delta_1} &\equiv 0 \pmod{N_1} \\ f_2(x) := x^{\delta_2} &\equiv 0 \pmod{N_2} \\ &\vdots \\ f_k(x) := x^{\delta_k} &\equiv 0 \pmod{N_k}\,. \end{aligned} \tag{3.4}$$

We would like to determine all solutions x_0 of this system with $|x_0| < N_1 = p_1^{\delta_1}$. An application of Theorem 2.3.9 to a single equation $f_i(x) \equiv 0 \pmod{N_i}$ efficiently yields all solutions x_0 with $|x_0| < (N_i)^{\frac{1}{\delta_i}} = p_i$. Furthermore, each multiple of p_i is a solution of $f_i(x) \equiv 0 \pmod{N_i}$. Thus, if x_0 is a multiple of $\prod_{i=1}^k p_i$, then x_0 is a common zero of all the polynomials.
Let $\delta := \operatorname{lcm}\{\delta_i, i = 1, \ldots, k\}$. We apply the same method as in the proof of Theorem 3.2.2 to the polynomial equations in system (3.4). Namely, we take their $\frac{\delta}{\delta_i}$-th powers and combine them by Chinese Remaindering (Theorem 2.2.13). This gives us an equation $f(x) \equiv x^\delta \pmod{M}$ with $M := \prod_{i=1}^k N_i^{\frac{\delta}{\delta_i}} = \prod_{i=1}^k p_i^\delta$ with the same roots as in (3.4).
We assume that $M^{\frac{1}{\delta}} < N_1$. Otherwise $M^{\frac{1}{\delta}} \geq N_1 > |x_0|$, i.e. the condition of Theorem 3.2.2 is fulfilled and there is nothing to be shown. Therefore, let $\epsilon > 0$ such that $M^{\frac{1}{\delta}+\epsilon} < N_1$. Suppose now we could calculate all simultaneous solutions x_0 of the system such that $|x_0| < M^{\frac{1}{\delta}+\epsilon} = (\prod_{i=1}^k p_i)^{1+\delta\epsilon}$. Since we know that every integer multiple of $\prod_{i=1}^k p_i$ is a solution of (3.4), the number of roots is roughly $2(\prod_{i=1}^k p_i)^{\delta\epsilon}$. This implies that we have exponentially many roots x_0 with $|x_0| < M^{\frac{1}{\delta}+\epsilon}$, which we cannot even output in polynomial time. Consequently, there is no polynomial time algorithm that improves upon the exponent in the condition $|x_0| < M^{\frac{1}{\delta}}$ of Theorem 3.2.2.

3.2.2 An Example

A typical example in which polynomially related messages occur is an RSA broadcast scenario. Assume a user wants to broadcast a message m to k different users using an RSA encryption scheme with public exponents $e_1, \ldots, e_k$ and coprime public moduli $N_1 < \ldots < N_k$. From the ciphertexts $c_1 \pmod{N_1}, \ldots, c_k \pmod{N_k}$ an attacker can compute the message m if m is smaller than the upper bound given in Theorem 3.2.2. He sets $f_i(x) := x^{e_i} - c_i \pmod{N_i}$ and applies Theorem 3.2.2.
In order to avoid sending various encryptions of the same message, a user might add some randomness r_i and then encrypt the linearly related messages $(m + r_i)$, $i = 1, \ldots, k$,

instead of m. However, if the attacker gets to know the randomness, he can calculate $F_i(x) := f_i(x + r_i) \pmod{N_i}$ and analyze the system of equations $F_i(x) \equiv 0 \pmod{N_i}$, $i = 1, \ldots, k$. As degree, modulus and leading coefficient are the same for $F_i(x)$ and $f_i(x)$, the upper bound on m up to which m can be recovered efficiently also remains unchanged. More generally, taking polynomially related messages instead of linearly related ones, the degree of $F_i(x)$, $i = 1, \ldots, k$, changes from e_i to $e_i\gamma_i$, where γ_i is the degree of the known polynomial relation.

Corollary 3.2.5
Let $k \in \mathbb{N}$, (N_i, e_i), $i = 1, \ldots, k$, be RSA public keys with $N_1 < N_2 < \ldots < N_k$ and coprime N_i. Furthermore, let $m \in \mathbb{Z}_{N_1}$ and let $g_i(x) \in \mathbb{Z}[x]$ be polynomials of degree $\gamma_i \in \mathbb{N}$ with $a_{i\gamma_i}$ being the coefficient of x^{γ_i} for $i = 1, \ldots, k$. Let $c_1, \ldots, c_k$ be the RSA-encryptions of $g_i(m)$ under the public key (N_i, e_i). Define $\delta_i := e_i\gamma_i$ and $M := \prod_{i=1}^{k} N_i^{\frac{\delta}{\delta_i}}$ with $\delta := lcm\{\delta_i, i = 1, \ldots, k\}$.
Then an adversary can recover the message m in time $\mathcal{O}(\delta^6 \log^2 M)$ provided that

$$\sum_{i=1}^{k} \frac{1}{\delta_i} \geq 1\,.$$

Proof: Without loss of generality we assume that all $a_{i\gamma_i}$ are invertible modulo N_i. (Otherwise $\gcd(a_{i\gamma_i}, N_i)$ and $\frac{N_i}{\gcd(a_{i\gamma_i}, N_i)}$ will give us the factorization of N_i for at least one $i \in \{1, \ldots, k\}$. We can then compute m modulo the prime factors. This can be done efficiently if δ_i is polynomial in $\log(N_i)$ as explained in the introduction of this chapter.)
We are looking for a solution m of $f_i(x) := g_i(x)^{e_i} - c_i \equiv 0 \pmod{N_i}$, $i = 1, \ldots, k$. However, the polynomials $f_i(x)$ are not necessarily monic. Therefore, we modify them slightly to be able to apply Corollary 3.2.3. Let $F_i(x) := a_{i\gamma_i}^{-e_i}\big(g_i(x)^{e_i} - c_i\big) \pmod{N_i}$, $i = 1, \ldots, k$. Hence, $F_i(x)$ is a monic polynomial of degree $\delta_i = e_i\gamma_i$. The corollary then directly follows as an application of Corollary 3.2.3. ■

Chapter 4

Basic Approaches for Solving Systems of Multivariate Polynomial Equations

Many problems occurring in the context of cryptography cannot be described as polynomial equations in one unknown but comprise several unknowns. Take, for example, the problem of factoring, i. e. given $N \in \mathbb{N}$ which is the product of two large primes p and q, determine p and q. We can easily transfer the problem of finding p and q into the problem of finding the non-trivial roots of a multivariate polynomial f: Define $f(x, y) := N - xy \in \mathbb{Z}[x, y]$. Then $f(p, q) = 0$.
In a more general setting, we are not only given one equation but several ones. The following three chapters deal with the problem of solving *systems of multivariate polynomial equations* (SMPE-problem), either over the integers (SIMPE-problem) or as modular systems (SMMPE-problem). The case of systems of modular equations can be divided up further into systems with a common modulus (SMMPE1) and systems with mutually coprime moduli (SMMPE2). That is, we consider the following problems:

Definition 4.0.6 (SIMPE-problem)
Assume $k \in \mathbb{N}$ and $f_1(x_1, \ldots, x_l), \ldots, f_k(x_1, \ldots, x_l)$ to be polynomials of degree $\delta_1, \ldots, \delta_k$ in $\mathbb{Z}[x_1, \ldots, x_l]$, respectively. Let

$$\begin{array}{rcl} f_1(x_1, \ldots, x_l) & = & 0 \\ f_2(x_1, \ldots, x_l) & = & 0 \\ & \vdots & \\ f_k(x_1, \ldots, x_l) & = & 0 \end{array} \tag{4.1}$$

be a system of multivariate polynomial equations.

Let $X_i \in \mathbb{R}$, $i = 1, \ldots, l$. Find all common roots $(\bar{x}_1, \ldots, \bar{x}_l)$ of (4.1) with size $|\bar{x}_i| \le X_i$.

and

Definition 4.0.7 (SMMPE-problem)
Assume $k \in \mathbb{N}$, $\delta_1, \dots, \delta_k \in \mathbb{N}$, and let $N_1, \dots, N_k \in \mathbb{N}$. Let $f_1(x_1, \dots, x_l), \dots, f_k(x_1, \dots, x_l)$ be polynomials of degree $\delta_1, \dots, \delta_k$ in $\mathbb{Z}_{N_1}[x_1, \dots, x_l], \dots, \mathbb{Z}_{N_k}[x_1, \dots, x_l]$, respectively. Let

$$\begin{aligned} f_1(x_1, \dots, x_l) &\equiv 0 \pmod{N_1} \\ f_2(x_1, \dots, x_l) &\equiv 0 \pmod{N_2} \\ &\vdots \\ f_k(x_1, \dots, x_l) &\equiv 0 \pmod{N_k} \end{aligned} \tag{4.2}$$

be a system of multivariate polynomial equations.

Let $X_i < N_i$, $X_i \in \mathbb{R}$, $i = 1, \dots, l$. Find all common roots $(\bar{x}_1, \dots, \bar{x}_l)$ of (4.2) with size $|\bar{x}_i| \leq X_i$.
If $N_1 = \ldots = N_k$, then we denote the problem by SMMPE1; if the N_i are mutually coprime, we assume $N_1 < N_2 < \ldots < N_k$ and refer to the problem as SMMPE2.

Our goal is to analyze up to which size of the upper bounds $X_1, \dots, X_l$ we can determine the solutions $\bar{x}_1, \dots, \bar{x}_l$ of either system of equations efficiently. In contrast to the univariate case (comp. Chapter 3), there are no algorithms working in general for multivariate equations over the integers either. That is why we are interested in analyzing the integer as well as the modular case. The analysis will be divided into three parts.

In the following two chapters we adapt Coppersmith's algorithm to be used to analyze systems of equations in the modular (Chapter 5) and in the integer case (Chapter 6). The rest of the current chapter deals with simpler approaches to solve systems of multivariate equations. Depending on the structure of the equations, a system can be modeled as a lattice such that a short vector in this lattice directly reveals the solution. The result is, thus, obtained by a simple lattice reduction. Before generally stating some criteria on which this analysis is useful, we treat the examples of related messages RSA [CFPR96] and implicit factoring [MR09].

4.1 RSA with Related Messages and Implicit Relations

We reconsider the problem of RSA with related messages. Assume two messages m_1 and m_2 are encrypted with respect to the same modulus N. The corresponding equations with solutions (m_1, m_2) are $f_1(x_1) := x_1^e - c_1 \equiv 0 \pmod{N}$ and $f_2(x_2) := x_2^e - c_2 \equiv 0 \pmod{N}$. In Example 3.1.2 the relation between m_1 and m_2 was explicit, namely, $m_2 \equiv p(m_1) \pmod{N}$. Thus, the system could be transformed into a system of two univariate equations $f_1(x_1) \equiv 0 \pmod{N}$ and $f_2(p(x_1)) \equiv 0 \pmod{N}$.
In [CFPR96] implicit polynomial relations $p(m_1, m_2) \equiv 0 \pmod{N}$ are considered as well. With such relations it is no longer possible to just substitute x_2 by a polynomial in x_1. Therefore, Don Coppersmith et al. add a further step in the method described in Example 3.1.2. As m_2 can no longer be directly substituted in $m_2^e - c_2 \equiv 0 \pmod{N}$, we regard

a system of three polynomial equations in two unknowns x_1 and x_2 corresponding to the solutions m_1 and m_2:

$$\begin{aligned} f_1(x_1) &\equiv x_1^e - c_1 \pmod N \\ f_2(x_2) &\equiv x_2^e - c_2 \pmod N \\ p(x_1, x_2) &\equiv 0 \pmod N\,. \end{aligned}$$

To get two polynomials in the same unknown, the resultant (compare Definition 2.2.9) of $f_1(x_1)$ and $p(x_1, x_2)$ with regard to x_1 is computed. This results in a polynomial $r(x_2)$ in the unknown x_2. We now have two polynomials in one unknown like in case of explicit relations, and (in most cases) we can compute $\gcd(r(x_2), f_2(x_2)) \equiv x_2 - m_2 \pmod N$, which gives us m_2. Plugging this value into $p(x_1, x_2)$, we get two polynomials $p(x_1, m_2)$ and $f_1(x_1)$ in the unknown x_1 and can proceed as before to determine m_1.
An alternative way to achieve this result is to compute a Groebner basis (compare Definition 2.2.11) of $\mathcal{F} = \{f_1(x_1), f_2(x_2), p(x_1, x_2)\}$. In most cases the Groebner basis contains the polynomials $x_1 - m_1$ and $x_2 - m_2$. From these polynomials we can immediately derive our solutions m_1 and m_2. Both variants of the attack can theoretically be performed with equations of any degree. However, the complexity of the attack is polynomial in the degree of the equations. Therefore, it is only efficient for equations of small degree. Commonly used moduli like $e = 3$ or $e = 2^{16} + 1$ fit into the framework of the attack.
Resultant or Groebner basis computations can be applied to solve nearly any two equations with the same modulus. This implies that it is usually sufficient to have two equations $f_1(x) \equiv 0 \pmod N$ and $f_2(x) \equiv 0 \pmod N$ in order to recover the common solutions as explained in Section 3.1.
The problem of RSA with related messages can be generalized to an arbitrary number of k messages. If the relations between these messages can be expressed explicitly such that m_i only depends on $m_1, \dots, m_{i-1}$, we can restrict our considerations to the first two equations and their common relation. Then we can perform the same analysis as described above. This gives us m_1 and m_2. Iteratively, any further message m_i, $i = 3, \dots, k$, can be calculated from $m_1, \dots, m_{i-1}$ by its explicit description.
A natural generalization, thus, includes only one implicit relation between the messages. Suppose we have the following system of $k + 1 \in \mathbb{N}$ equations

$$\begin{aligned} f_1(x_1) := x_1^e - c_1 &\equiv 0 \pmod N \\ &\vdots \\ f_k(x_k) := x_k^e - c_k &\equiv 0 \pmod N \\ p(x_1, \dots, x_k) &\equiv 0 \pmod N\,. \end{aligned}$$

As before, in most cases we can either compute the Groebner basis of all polynomials and obtain the answer $[x_1 - m_1, \dots, x_k - m_k]$. Or, we set

$$g_0(x_1, \dots, x_k) := p(x_1, \dots, x_k)$$

and iteratively compute

$$g_i(x_{i+1},\dots,x_k) \;:=\; res_{x_i}(g_{i-1}(x_i,\dots,x_k), f_i(x_i))\,.$$

After k iterations one gets $g_{k-1}(x_k)$. Computing the greatest common divisor of $g_{k-1}(x_k)$ and $f_k(x_k)$ in most cases leads to $\gcd(g_{k-1}(x_k), f_k(x_k)) = x_k - m_k$. Recursively, for $i = k-1,\dots,1$, we can substitute $x_{i+1},\dots,x_k$ by $m_{i+1},\dots,m_k$, and calculate

$$\gcd(g_{i-1}(x_i, m_{i+1},\dots,m_k), f_i(x_i)) \;=\; x_i - m_i$$

to get m_i until we have m_1.

The techniques used to analyze this problem are basically the same techniques described in Section 2.2. The only difference is that the coefficients here are elements of $\mathbb{Z}_N$. Therefore, we have to use a definition and computation of Groebner bases adapted to coefficients in any ring (compare [Pau07]). Then we can determine all solutions via Groebner basis computations if the Groebner basis contains univariate linear polynomials in the different variables. There are no constraints concerning the solutions we can determine. This is an advantage of this method in contrast to the methods we will present in the following. In those methods, lattice reduction techniques are used. This leads to size constraints on the solutions which we can determine this way.
Practical problems, however, rarely correspond to systems of equations such that a Groebner basis directly reveals the solution. Instead, a Groebner basis computation leads to a different, usually simpler system which is still too complicated to derive solutions from. Note further that the complexity of Groebner basis computations is exponential in the number of variables. Hence, this approach is not efficient in general (compare Section 2.2). This is a drawback compared to the lattice-based methods we will use.
We will analyze the problem of RSA with related messages with the help of Coppersmith's method in Chapter 5.2. Now, we will continue with an example of a non zero dimensional system of which we can determine solutions of a certain size.

4.2 The Problem of Implicit Factoring with Shared Least Significant Bits

In this section we again regard the problem of factoring, that is, given a composite integer $N_0 = p_0q_0$, compute p_0 and q_0. As this problem seems to be difficult to solve in general, we use an additional oracle. In contrast to previous works like [RS85, Mau96, Cop96a], we highly restrict the power of the oracle. Namely, we allow for an oracle that on input an RSA modulus $N_0 = p_0q_0$ outputs another different RSA modulus $N_1 = p_1q_1$ such that p_0 and p_1 share their t least significant bits. Moreover, we assume for notational simplicity that the bitsize of p_1 is equal to the bitsize of p_0 and the bitsize of q_1 is equal to the bitsize of q_0.
Thus, as opposed to an oracle that *explicitly* outputs bits of the prime factor p_0, we only have an oracle that *implicitly* gives information about the bits of p_0. Intuitively, since

N_1 is a hard to factor RSA modulus, it should not be possible to extract this *implicit* information. We show that this intuition is false. Namely, we show that the link of the factorization problems N_0 and N_1 gives rise to an efficient factorization algorithm provided that t is large enough.

More precisely, let q_0 and q_1 be α-bit numbers. Then our lattice-based algorithm provably factors N_0, N_1 with $N_0 \neq N_1$ in quadratic time whenever $t > 2(\alpha + 1)$. In order to give a numerical example: Let N_0, N_1 have 750-bit p_0, p_1 and 250-bit q_0, q_1. Then the factorization of N_0, N_1 can be efficiently found provided that p_0, p_1 share more than 502 least significant bits. The bound $t > 2(\alpha + 1)$ implies that our first result works only for imbalanced RSA moduli. Namely, the prime factors p_i have to have bitsizes larger than twice the bitsizes of the q_i.

Using more than one oracle query, we can further improve upon the bound on t. In case of k queries, we obtain $N_1, \ldots, N_k$ different RSA moduli such that all p_i share the t least significant bits. This gives rise to a lattice attack with a $(k+1)$-dimensional lattice L having a short vector $\mathbf{q} = (q_0, q_1, \ldots, q_k)$ that immediately yields the factorization of all $N_0, N_1, \ldots, N_k$. For constant k, our algorithm runs in time polynomial in the bitsize of the RSA moduli. As opposed to our first result, in the general case we are not able to prove that our target vector $\mathbf{q}$ is a shortest vector in the lattice L. Thus, we leave this as a heuristic assumption. This heuristic is supported by a counting argument and by experimental results that demonstrate that we are almost always able to efficiently find the factorization.

Moreover, when putting k queries for RSA moduli with α-bit q_i that share t least significant bits of the p_i, we improve our bound to $t \geq \frac{k+1}{k}\alpha$. Hence, for a larger number k of queries our bound converges to $t \geq \alpha$, which means that the p_i should at least coincide on α bits, where α is the bitsize of the q_i. In case the two prime factors have the same bitsize, this result tells us that $N_0 = p_0 q_0, \ldots, N_k = p_0 q_k$ with the same p_0 can efficiently be factored, which is trivially true by greatest common divisor computations. On the other hand, our result is non-trivial whenever the bitsizes are not balanced.

If we do not restrict ourselves to polynomial running time, then we can easily adapt our method to factor balanced RSA moduli as well. All that we have to do is to determine a small quantity of the bits of q_i by brute force search. Using these bits, we can apply the previous method in order to determine at least half of the bits of all q_i. The complete factorization of *all* RSA moduli N_i is then retrieved by the aforementioned lattice-based algorithm of Coppersmith [Cop96a].

4.2.1 Implicit Factoring of Two RSA Moduli

Let us start with the analysis of implicit factoring with only one oracle query. Assume that we are given two different RSA moduli $N_0 = p_0 q_0, N_1 = p_1 q_1$, where p_0, p_1 coincide on the t least significant bits. That is, $p_0 = p + 2^t \tilde{p}_0$ and $p_1 = p + 2^t \tilde{p}_1$ for a common value p that is *unknown* to us. Can we use the information that the prime factors of N_0 and N_1 share their t least significant bits without knowing these bits *explicitly*? That is, can we factor N_0, N_1 given only *implicit* information about one of the factors?

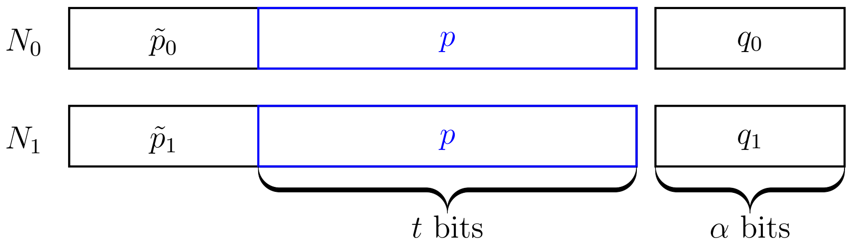

Figure 4.1: Illustration of N_0 and N_1 for implicit factoring in terms of bits.

In this section, we will answer this question in the affirmative. We will show that there is an algorithm that recovers the factorization of N_0 and N_1 in quadratic time provided that t is sufficiently large.
We start with

$$\begin{aligned}(p + 2^t\tilde{p}_0)q_0 &= N_0\\ (p + 2^t\tilde{p}_1)q_1 &= N_1\,.\end{aligned}$$

These two equations contain five unknowns $p, \tilde{p}_0, \tilde{p}_1, q_0$ and q_1. By reducing both equations modulo 2^t, we can eliminate the two unknowns $\tilde{p}_0, \tilde{p}_1$ and obtain

$$\begin{aligned}pq_0 &\equiv N_0 \pmod{2^t}\\ pq_1 &\equiv N_1 \pmod{2^t}\,.\end{aligned}$$

Since q_0, q_1 are odd, we can solve both equations for p. This leaves us with $\frac{N_0}{q_0} \equiv \frac{N_1}{q_1} \pmod{2^t}$, which we write in form of the *linear* equation

$$(N_0^{-1}N_1)q_0 - q_1 \equiv 0 \pmod{2^t}\,. \tag{4.3}$$

The set of solutions

$$L = \{(x_0, x_1) \in \mathbb{Z}^2 \mid (N_0^{-1}N_1)x_0 - x_1 \equiv 0 \pmod{2^t}\}$$

forms an additive, discrete subgroup of $\mathbb{Z}^2$. Thus, L is a 2-dimensional integer lattice. L is spanned by the row vectors of the basis matrix

$$\mathbf{B_L} := \begin{pmatrix} 1 & N_0^{-1}N_1 \\ 0 & 2^t \end{pmatrix}.$$

This equivalence has been proven in Section 2.3. Notice that by equation (4.3), we have $(q_0, q_1) \in L$. If we were able to find this vector in L, then we could factor N_0, N_1 easily. Let us first provide some intuition on which condition the vector $\mathbf{q} = (q_0, q_1)$ is a short vector in L. We know that an upper bound on the length of a shortest vector is given by the Minkowski bound $\sqrt{2}\det(L)^{\frac{1}{2}} = \sqrt{2} \cdot 2^{\frac{t}{2}}$.
Since we assumed that q_0, q_1 are α-bit primes, we have $q_0, q_1 \le 2^\alpha$. If α is sufficiently small,

then $||\mathbf{q}||$ is smaller than the Minkowski bound and, therefore, we can expect that $\mathbf{q}$ is among the shortest vectors in L. This happens if

$$||\mathbf{q}|| \leq \sqrt{2} \cdot 2^{\alpha} \leq \sqrt{2} \cdot 2^{\frac{t}{2}}.$$

So if $t \geq 2\alpha$ we expect that $\mathbf{q}$ is a short vector in L. We can find a shortest vector in L using Gaussian reduction on the lattice basis $\mathbf{B_L}$ in time $\mathcal{O}(\log^2(2^t)) = \mathcal{O}(\log^2(\min\{N_0, N_1\}))$. Hence, on the heuristic assumption that $\mathbf{q} = (q_0, q_1)$ is a shortest vector in L, we can factor N_0, N_1 in quadratic time. On a slightly more restrictive condition, we can completely remove the heuristic assumption.

Theorem 4.2.1
Let $N_0 = p_0 q_0, N_1 = p_1 q_1$ be two different RSA moduli with α-bit q_i. Suppose that p_0, p_1 share at least $t > 2(\alpha + 1)$ bits. Then N_0 and N_1 can be factored in quadratic time.

PROOF: Let

$$\mathbf{B_L} = \begin{pmatrix} 1 & N_0^{-1} N_1 \\ 0 & 2^t \end{pmatrix}$$

be the lattice basis defined as before.
The basis matrix $\mathbf{B_L}$ spans a lattice L with a shortest vector $\mathbf{v}$ that satisfies

$$||\mathbf{v}|| \leq \sqrt{2} \det(L)^{\frac{1}{2}} = 2^{\frac{t+1}{2}}.$$

Performing Gaussian reduction on $\mathbf{B_L}$, we get an equivalent basis $\mathbf{B} = \begin{pmatrix} \mathbf{b_1} \\ \mathbf{b_2} \end{pmatrix}$ such that

$$||\mathbf{b_1}|| = \lambda_1(L) \text{ and } ||\mathbf{b_2}|| = \lambda_2(L)\,.$$

Our goal is to show that $\mathbf{b_1} = \pm\mathbf{q} = \pm(q_0, q_1)$ which is sufficient for factoring N_0 and N_1. As L is of full rank, by Hadamard's inequality we have

$$||\mathbf{b_1}||\,||\mathbf{b_2}|| \geq \det(L)\,.$$

This implies

$$||\mathbf{b_2}|| \geq \frac{\det(L)}{||\mathbf{b_1}||} = \frac{\det(L)}{\lambda_1(L)}\,.$$

Substituting $\det(L) = 2^t$ and using $\lambda_1(L) \leq 2^{\frac{t+1}{2}}$ leads to

$$||\mathbf{b_2}|| \geq \frac{2^t}{2^{\frac{t+1}{2}}} = 2^{\frac{t-1}{2}}\,.$$

This implies for any lattice vector $\mathbf{0} \neq \mathbf{v} = a_1\mathbf{b_1} + a_2\mathbf{b_2}$ with $||\mathbf{v}|| < 2^{\frac{t-1}{2}}$ that $a_2 = 0$, as otherwise $\lambda_2(L) \leq ||\mathbf{v}|| < ||\mathbf{b_2}||$, which contradicts the optimality of $\mathbf{b_2}$ from Theorem 2.3.3. Thus, every $\mathbf{v} \neq \mathbf{0}$ with $||\mathbf{v}|| < 2^{\frac{t-1}{2}}$ is a multiple of $\mathbf{b_1}$. Notice that $\mathbf{q} = (q_0, q_1) \in L$ fulfills $||\mathbf{q}|| = \sqrt{2} \cdot 2^{\alpha} = 2^{\frac{2\alpha+1}{2}}$. Consequently, we have $||\mathbf{q}|| < ||\mathbf{b_2}||$ if

$$2^{\frac{2\alpha+1}{2}} < 2^{\frac{t-1}{2}} \Leftrightarrow 2(\alpha + 1) < t\,.$$

Therefore, we get $\mathbf{q} = a\mathbf{b_1}$ with $a \in \mathbb{Z} \setminus \{0\}$. Let $\mathbf{b_1} = (b_{11}, b_{12})$, then $\gcd(q_0, q_1) = \gcd(ab_{11}, ab_{12}) \geq a$. But q_0, q_1 are primes and, without loss of generality, $q_0 \neq q_1$, since otherwise we can factor N_0, N_1 by computing $\gcd(N_0, N_1)$. Therefore, by the minimality of $\mathbf{b_1}$, $|a| = 1$ and we obtain $\mathbf{q} = \pm\mathbf{b_1}$, which completes the factorization.
The running time of the factorization is determined by the running time of the Gaussian reduction, which can be performed in $\mathcal{O}(t^2) = \mathcal{O}(\log^2(\min\{N_0, N_1\}))$ steps. ■

4.2.2 Implicit Factoring of k RSA Moduli

The approach from the previous section can be generalized to an arbitrary fixed number k of oracle queries. This gives us $k+1$ different RSA moduli

$$\begin{aligned} N_0 &= (p + 2^t\tilde{p}_0)q_0 \\ &\vdots \\ N_k &= (p + 2^t\tilde{p}_k)q_k \end{aligned} \tag{4.4}$$

with α-bit q_i.
We transform the system of equations into a system of $k+1$ equations modulo 2^t

$$\begin{aligned} pq_0 - N_0 &\equiv 0 \pmod{2^t} \\ &\vdots \\ pq_k - N_k &\equiv 0 \pmod{2^t} \end{aligned}$$

in $k+2$ variables.
Analogous to the case of two equations, we solve each equation for p. This can be done because all the q_i are odd. Thus, we get $\frac{N_0}{q_0} = \frac{N_i}{q_i} \pmod{2^t}$ for $i = 1, \ldots, k$. Writing this as k linear equations gives us:

$$\begin{aligned} N_0^{-1}N_1q_0 - q_1 &\equiv 0 \pmod{2^t} \\ &\vdots \\ N_0^{-1}N_kq_0 - q_k &\equiv 0 \pmod{2^t}. \end{aligned}$$

With the same arguments as in the preceding section, the set

$$L = \{(x_0, x_1, \ldots, x_k) \in \mathbb{Z}^{k+1} \mid N_0^{-1}N_ix_0 - x_i \equiv 0 \pmod{2^t} \text{ for all } i = 1, \ldots, k\}$$

forms a lattice. This lattice L is spanned by the row vectors of the following basis matrix

$$\mathbf{B_L} = \begin{pmatrix} 1 & N_0^{-1}N_1 & \cdots & & N_0^{-1}N_k \\ 0 & 2^t & 0 & \cdots & 0 \\ 0 & 0 & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & 0 & \cdots & 0 & 2^t \end{pmatrix}.$$

Note that $\mathbf{q} = (q_0, q_1, \dots, q_k) \in L$ has norm $||\mathbf{q}|| \leq \sqrt{k+1} \cdot 2^{\alpha}$. We would like to have $||\mathbf{q}|| = \lambda_1(L)$ as in Section 4.2.1. The length $\lambda_1(L)$ of a shortest vector in L is bounded by

$$\lambda_1(L) \leq \sqrt{k+1} \cdot (\det(L))^{\frac{1}{k+1}} = \sqrt{k+1} \cdot (2^{tk})^{\frac{1}{k+1}}.$$

Thus, if $\mathbf{q}$ is indeed a shortest vector, then

$$||\mathbf{q}|| = \sqrt{k+1} \cdot 2^{\alpha} < \sqrt{k+1} \cdot 2^{t\frac{k}{k+1}}. \tag{4.5}$$

This implies the condition $t > \frac{k+1}{k}\alpha$. We make the following heuristic assumption.

Assumption 4.2.2
Let $N_0, N_1, \dots, N_k$ be as defined in equation (4.4) *with $t > \frac{k+1}{k}\alpha$. Further, let $\mathbf{b_1}$ be a shortest vector in L. Then $\mathbf{b_1} = \pm(q_0, q_1, \dots, q_k)$.*

Theorem 4.2.3
Let $N_0, \dots, N_k$ be as defined in the system of equations (4.4) *with $t > \frac{k+1}{k}\alpha$. Under Assumption 4.2.2, we can find the factorization of all $N_0, N_1, \dots, N_k$ in time polynomial in $((k+1)^{\frac{k+1}{2}}, \max_i\{\log N_i\})$.*

We show the validity of Assumption 4.2.2 experimentally in Section 4.2.5.
The running time is determined by the time to compute a shortest vector in L [Kan87, Hel85]. This implies that for any lattice L of rank $k+1$ we can compute the factorization of all N_i in time polynomial in their bitsize if $(k+1)^{\frac{k+1}{2}} = \text{poly}(\max_i\{\log N_i\})$, that is, especially, if the lattice has fixed rank $k+1$.
For large k, our bound converges to $t \geq \alpha$. This means that the amount t of common least significant bits has to be at least as large as the bitsize of the q_i. In turn, this implies that our result only applies to RSA moduli with different bitsizes of p_i and q_i. On the other hand, this is the best result that we could hope for in our algorithm. Notice that we construct the values of the q_i by solving equations modulo 2^t. Thus, we can fully recover the q_i only if their bitsize α is smaller than t. In the subsequent section, we will overcome this problem by avoiding the full recovery of all q_i, which in turn leads to an algorithm for balanced RSA moduli.

Remark: All of our results still hold if 2^t is replaced by an arbitrary modulus $M \geq 2^t$. We used a power of two only to illustrate our results in terms of bits.

4.2.3 Implicit Factoring of Balanced RSA Moduli

We slightly adapt the method from Section 4.2.2 in order to factor balanced n-bit integers, i. e. $N_i = p_i q_i$ such that p_i and q_i have bitsize $\frac{n}{2}$ each. The modification mainly incorporates a small brute force search on the most significant bits of the q_i.
Assume that we are given $k+1$ RSA moduli as in (4.4). From these moduli we derive k

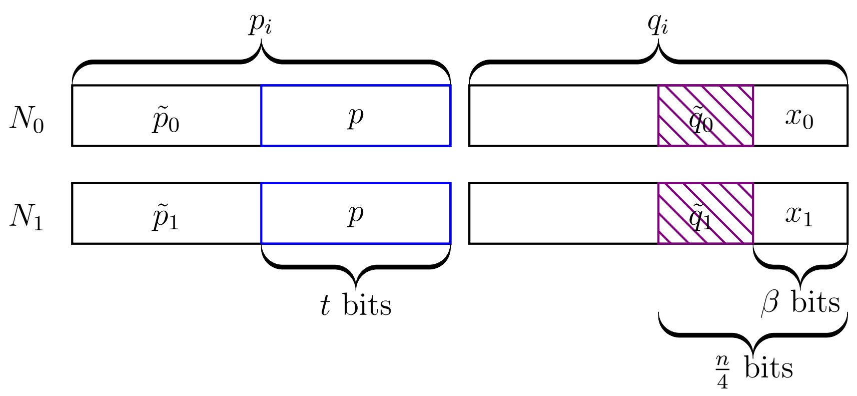

Figure 4.2: Illustration of balanced N_0 and N_1 for implicit factoring in terms of bits.

linear equations in $k+1$ variables:

$$\begin{aligned} N_0^{-1}N_1q_0 - q_1 &\equiv 0 \pmod{2^t} \\ &\vdots \\ N_0^{-1}N_kq_0 - q_k &\equiv 0 \pmod{2^t}. \end{aligned}$$

The bitsize of the q_i is now fixed to $\alpha = \frac{n}{2}$, which is equal to the bitsize of the p_i, i. e. now the number t of bits on which the p_i coincide has to satisfy $t \le \alpha$. In the trivial case of $t = \alpha = \frac{n}{2}$ we can directly factor the N_i via greatest common divisor computations as then $p_i = p$ for $i = 0, \ldots, k$.
Thus, we only consider $t < \frac{n}{2}$. With a slight modification of the method in Section 4.2.2, we compute all $q_i \pmod{2^t}$. Since $t < \frac{n}{2}$, this does not give us the q_i directly, but only their t least significant bits. But if $t \ge \frac{n}{4}$, we can use Theorem 2.3.12 for finding the full factorization of each N_i in polynomial time. In order to minimize the time complexity, we assume $t = \frac{n}{4}$ throughout this section.
To apply Theorem 4.2.3 of Section 4.2.2 the bitsize of the unknown part of the q_i has to be smaller than $\frac{k}{k+1}t$. Thus, we have to guess roughly $\frac{1}{k+1} \cdot t = \frac{n}{4(k+1)}$ bits for each q_i. Since we consider $k+1$ moduli, we have to guess a total number of $\frac{n}{4}$ bits. Notice that this is the same amount of bits as for guessing one half of the bits of *one* q_j, which in turn allows to efficiently find this q_j using Theorem 2.3.12. With a total amount of $\frac{n}{4}$ bits, however, our algorithm will allow us to efficiently find *all* q_i, $i = 0, \ldots, k$.
Let us describe our modification more precisely. We split $q_i \pmod{2^{\frac{n}{4}}}$ into $2^\beta \tilde{q}_i + x_i \pmod{2^{\frac{n}{4}}}$. The number β depends on the number of oracle calls k such that the condition $\beta < \frac{k}{k+1} \cdot \frac{n}{4}$ holds. Therefore, we choose β to be the largest integer smaller than $\frac{kn}{4(k+1)}$. This implies that the $x_i \le 2^\beta$ are small enough to be determined analogously as in Section 4.2.2, provided that the $\tilde{q}_i$ are known. In practice we can guess an amount of $\frac{n}{4(k+1)}$ bits for determining each $\tilde{q}_i$, or we can find these bits by other means, e.g. by side-channel attacks.

Suppose now that the $\tilde{q}_i$ are given for each i. We obtain the following set of equations

$$\begin{aligned} N_0^{-1}N_1x_0 - x_1 &\equiv 2^\beta(\tilde{q}_1 - N_0^{-1}N_1\tilde{q}_0) \pmod{2^{\frac{n}{4}}} \\ &\vdots \\ N_0^{-1}N_kx_0 - x_k &\equiv 2^\beta(\tilde{q}_k - N_0^{-1}N_k\tilde{q}_0) \pmod{2^{\frac{n}{4}}}\,. \end{aligned} \tag{4.6}$$

Let $c_i = 2^\beta(\tilde{q}_i - N_0^{-1}N_i\tilde{q}_0)$, $i = 1, \dots, k$, denote the known right-hand terms. In contrast to Section 4.2.2, the equations (4.6) that we have to solve are inhomogeneous. Let us first consider the lattice L that consists of the homogeneous solutions

$$L = \{(x_0, x_1, \dots, x_k) \in \mathbb{Z}^{k+1} \mid N_0^{-1}N_ix_0 - x_i \equiv 0 \pmod{2^{\frac{n}{4}}}, i = 1, \dots, k\}\,.$$

L is spanned by the rows of the following basis matrix

$$\mathbf{B_L} = \begin{pmatrix} 1 & N_0^{-1}N_1 & \cdots & & N_0^{-1}N_k \\ 0 & 2^{\frac{n}{4}} & 0 & \cdots & 0 \\ 0 & 0 & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & 0 & \cdots & 0 & 2^{\frac{n}{4}} \end{pmatrix}.$$

Let $l_i \in \mathbb{Z}$ such that $N_0^{-1}N_ix_0 + l_i2^{\frac{n}{4}} = x_i + c_i$. Then we let

$$\mathbf{q}' \;:=\; (x_0, l_1, \dots, l_k)\mathbf{B_L} = (x_0, x_1 + c_1, \dots, x_k + c_k) \in L.$$

Moreover, if we define the target vector $\mathbf{c} := (0, c_1, \dots, c_k)$, then the distance between $\mathbf{q}'$ and $\mathbf{c}$ is

$$||\mathbf{q}' - \mathbf{c}|| = ||(x_0, x_1, \dots, x_k)|| \leq \sqrt{k+1}\cdot 2^\beta \leq \sqrt{k+1}\cdot 2^{\frac{kn}{4(k+1)}}.$$

This is the same bound that we achieved in Section 4.2.2 for the length of a shortest vector in equation (4.5) in the case of $t = \frac{n}{4}$. So instead of solving a shortest vector problem, we have to solve a closest vector problem in L with target vector $\mathbf{c}$. Closest vectors can be found in polynomial time for fixed lattice dimension $k + 1$ (see Blömer [Blö00]). We make the heuristic assumption that $\mathbf{q}'$ is indeed a closest vector to $\mathbf{c}$ in L.

Assumption 4.2.4
Let $N_0, N_1, \dots, N_k$ be as defined in equation (4.6) with $\beta < \frac{kn}{4(k+1)}$. Further, let $\mathbf{b_1}$ be a closest vector to $\mathbf{c}$ in L. Then $\mathbf{b_1} = \pm\mathbf{q}'$.

Theorem 4.2.5
Let $N_0, N_1, \dots, N_k$ be as defined in equation (4.6) with $\beta < \frac{kn}{4(k+1)}$. On Assumption 4.2.4, we can find the factorization of all $N_0, N_1, \dots, N_k$ in time $2^{\frac{n}{4}} \cdot \mathrm{poly}((k+1)!, \max_i\{\log N_i\})$.

The running time is determined by the time for guessing each $\tilde{q}_i$ and the time for finding a closest vector in L.
In the following section we have a closer look at the two heuristics from the previous sections, Assumption 4.2.2 and Assumption 4.2.4. We first give a counting argument that supports our heuristics and then demonstrate experimentally that our constructions work well in practice.

4.2.4 A Counting Argument that Supports our Assumptions

Recall that in Section 4.2.2, the lattice L consists of all solutions $\mathbf{q} = (q_0, q_1, \ldots, q_k)$ of the system of equations

$$\begin{aligned} N_0^{-1} N_1 q_0 &\equiv q_1 \pmod{2^t} \\ &\vdots \\ N_0^{-1} N_k q_0 &\equiv q_k \pmod{2^t}. \end{aligned} \tag{4.7}$$

As $\gcd(N_0^{-1}N_i, 2^t) = 1$ for any i, the mapping $f_i : x \mapsto N_0^{-1}N_i x \pmod{2^t}$ is bijective. Therefore, the value of q_0 uniquely determines the values of q_i, $i = 1, \ldots, k$.
In total, the system of equations has as many solutions as there are values to choose q_0 from, which is 2^t. Now suppose $q_0 \leq 2^{\frac{kt}{k+1}}$. How many vectors $\mathbf{q}$ do we have such that $q_i \leq 2^{\frac{kt}{k+1}}$ for all $i = 0, \ldots, k$ and, thus, $||\mathbf{q}|| \leq \sqrt{k+1} \cdot 2^{\frac{kt}{k+1}}$?
Assume for each $i = 1, ..., k$ that the value q_i is uniformly distributed in $\{0, \ldots, 2^t - 1\}$ and that the distributions of q_i and q_j are independent if $i \neq j$. Then the probability that $q_i \leq 2^{\frac{kt}{k+1}}$ is

$$\Pr\left(q_i \leq 2^{\frac{kt}{k+1}}\right) = \frac{2^{\frac{kt}{k+1}}}{2^t} = 2^{-\frac{t}{k+1}}.$$

Furthermore, the probability that $q_i \leq 2^{\frac{kt}{k+1}}$ for all $i = 1, \ldots, k$ is

$$\Pr\left(q_1 \leq 2^{\frac{kt}{k+1}}, \ldots, q_k \leq 2^{\frac{kt}{k+1}}\right) = \left(2^{-\frac{t}{k+1}}\right)^k = 2^{-\frac{kt}{k+1}}.$$

Consequently, the expected number of vectors $\mathbf{q}$ such that $q_i \leq 2^{\frac{kt}{k+1}}$ for all $i = 0, \ldots, k$ is $2^{\frac{kt}{k+1}} \cdot 2^{-\frac{kt}{k+1}} = 1$. Therefore, we expect that only one lattice vector, namely $\mathbf{q}$, is short enough to satisfy the Minkowski bound. Hence, we expect that $\pm\mathbf{q}$ is a unique shortest vector in L if its length is significantly below the bound $\sqrt{k+1} \cdot 2^{\frac{kt}{k+1}}$. This counting argument strongly supports our Assumption 4.2.2.

Remark: In order to analyze Assumption 4.2.4 we can argue in a completely analogous manner. The inhomogeneous character of the equations does not influence the fact that the q_i are uniquely determined by q_0.

4.2.5 Experiments

We verified our assumptions in practice by running experiments on a Core2 Duo 1.66GHz notebook. The attacks were implemented using Magma[1] Version 2.11. Instead of taking a lattice reduction algorithm which provably returns a basis with a shortest vector as first basis vector, we have used the LLL algorithm [LLL82], more precisely its L^2 version of Phong Nguyen and Damien Stehlé [NS05], which is implemented in Magma. Although by

[1]http://magma.maths.usyd.edu.au/magma/

LLL-reduction the first basis vector only approximates a shortest vector in a lattice, for our lattice bases with dimensions up to 100 LLL-reduction was sufficient. In nearly all cases the first basis vector was equal to the vector $\pm\mathbf{q} = \pm(q_0, q_1, \ldots, q_k)$, provided that we chose suitable attack parameters.
First, we considered the case of imbalanced RSA moduli from Theorem 4.2.3. We chose $N_i = (p + 2^t\tilde{p}_i)q_i$, $i = 0, \ldots, k$, of bitsize $n = 1000$ with varying bitsizes of q_i. For a fixed bitsize α of q_i and a fixed number k of moduli, we slightly played with the parameter t of common bits close to the bound $t \geq \frac{k+1}{k}\alpha$ in order to determine the minimal t for which our heuristic is valid.

bitsize α of the q_i	number of moduli $k+1$	bound $\frac{k+1}{k}\alpha$	number of shared bits t	success rate
250	3	375	377	0%
250	3	375	378	97%
350	10	389	390	0%
350	10	389	391	100%
400	100	405	409	0%
400	100	405	410	100%
440	50	449	452	16%
440	50	449	453	97%
480	100	485	491	38%
480	100	485	492	98%

Table 4.1: Attack for imbalanced RSA moduli

The running time of all experiments was below 10 seconds.
In Table 4.1, we called an experiment successful if the first basis vector $\mathbf{b}_1$ in our LLL-reduced basis was of the form $\mathbf{b_1} = \pm\mathbf{q} = \pm(q_0, q_1, \ldots, q_k)$, i.e. it satisfied Assumption 4.2.2. There were some cases where other basis vectors were of the form $\pm\mathbf{q}$, but we did not consider these cases as successful.
As one can see by the experimental results, Assumption 4.2.2 only works smoothly when our instances were a few extra bits beyond the bound of Theorem 4.2.3. This is not surprising since the counting argument from Section 4.2.4 tells us that we loose uniqueness of the shortest vector as we approach the theoretical bound. In practice, one could either slightly increase the number t of shared bits or the number k of oracle calls for making the attack work. Alternatively, one could also perform a small brute force search in a few bits.

Analogously, we made experiments with balanced RSA moduli to verify Assumption 4.2.4. Instead of computing closest vectors directly, we used the well-known standard embedding of a d-dimensional closest vector problem into an $(d+1)$-dimensional shortest vector problem ([MG02], Chapter 4.1), where the shortest vector is of the form $\mathbf{b_1} = (\mathbf{q}' - \mathbf{c}, c')$, c' constant. Since $\mathbf{c}$ and c' are known, this directly yields $\mathbf{q}'$ and, therefore, the factorization of all RSA moduli. For solving the shortest vector problem, we again used the LLL algo-

rithm.
As before we called an experiment successful if $\mathbf{b_1}$ was of the desired form, i.e. if Assumption 4.2.4 held. In our experiments we used 1000-bit N_i with a common share p of $t = 250$ bits.

number of moduli $k+1$	bound $\lceil \frac{n}{4(k+1)} \rceil$	bits known from q_i	success rate
3	84	85	74%
3	84	86	99%
10	25	26	20%
10	25	27	100%
50	5	8	46%
50	5	9	100%

Table 4.2: Attack for balanced 1000-bit N_i with 250 bits shared

All of our experiments ran in less than 10 seconds. Here we assumed that we know the required bits of each q_i, i.e. the running time does not include the factor for a brute-force search.
Similar to the experimental results in the imbalanced RSA case, our heuristic Assumption 4.2.4 works well in the balanced case, provided that we spend a few extra bits to the theoretical bound in order to enforce uniqueness of the closest vector.

4.3 The Problem of Implicit Factoring with Shared Most Significant Bits

In [MR09] we have introduced the problem of implicit factoring with respect to shared least significant bits. Jean-Charles Faugère, Raphaël Marinier and Guénaël Renault extended the analysis to the problem of implicit factoring with shared most significant bits [FMR09]. This extension is not straightforward as the non-shared bits of the larger factor can no longer be ignored. To see this, let us consider both variants of the factors in the case of two composite numbers N_0 and N_1.
Recall that in the case of shared least significant bits we regard the equations

$$\begin{aligned} \left(2^t\tilde{p}_0 + p\right) q_0 &= N_0 \\ \left(2^t\tilde{p}_1 + p\right) q_1 &= N_1 \,. \end{aligned} \tag{4.8}$$

Solving them for p and combining them, we obtain the equation

$$N_0q_1 - N_1q_0 - 2^t\left(\tilde{p}_0 - \tilde{p}_1\right) q_0q_1 = 0\,. \tag{4.9}$$

Then we take the equation modulo 2^t. By this, we can remove the unknowns $\tilde{p}_0$ and $\tilde{p}_1$. The remaining equation is equivalent to

$$N_0^{-1} N_1 q_0 \equiv q_1 \pmod{2^t}.$$

This linear equation can then be embedded into a lattice. If q_0 and q_1 are small enough (which is the case if $t > 2(\alpha+1)$ as we have seen in the previous section), we can determine them as solutions of a shortest vector problem.
In the case of shared most significant bits the situation changes. We are no longer able to just remove $\tilde{p}_0$ and $\tilde{p}_1$ by a modulo operation. In the case of shared most significant bits the equations have the following shape:

$$\begin{aligned} \left(\tilde{p}_0 + 2^{n-t-\alpha} p\right) q_0 &= N_0 \\ \left(\tilde{p}_1 + 2^{n-t-\alpha} p\right) q_1 &= N_1 . \end{aligned} \tag{4.10}$$

Analogously to the previous case, they can be solved for $2^{n-t-\alpha}p$ and combined:

$$\begin{aligned} N_1 q_0 - N_0 q_1 - (\tilde{p}_1 - \tilde{p}_0)\, q_0 q_1 &= 0 \\ \Leftrightarrow N_1 q_0 - N_0 q_1 &= (\tilde{p}_1 - \tilde{p}_0)\, q_0 q_1 . \end{aligned} \tag{4.11}$$

The variables $\tilde{p}_0$ and $\tilde{p}_1$ are still contained in the integer equation. However, they correspond to a monomial of size bounded by $2^{n+\alpha-t+1}$. If this is still small enough, we can embed the integer equation into a lattice L and determine it by calculating a shortest vector. We define the lattice L via a basis matrix

$$\mathbf{B_L} := \begin{pmatrix} \mathbf{b_1} \\ \mathbf{b_2} \end{pmatrix} = \begin{pmatrix} 2^{n-t+\frac{1}{2}} & 0 & N_1 \\ 0 & 2^{n-t+\frac{1}{2}} & -N_0 \end{pmatrix}. \tag{4.12}$$

The vector $\mathbf{q} := (2^{n-t+\frac{1}{2}} q_0, 2^{n-t+\frac{1}{2}} q_1, (\tilde{p}_1 - \tilde{p}_0) q_0 q_1) = q_0 \mathbf{b_1} + q_1 \mathbf{b_2}$ is contained in the lattice L. If it is a shortest vector, it can be determined via lattice reduction. Jean-Charles Faugère et al. show that $\mathbf{q}$ indeed is a shortest vector in L if $t > 2(\alpha + 1)$. This matches the bound we have obtained in the least significant bit case. The method can be extended to more than two composite integers N_i in a straightforward way. In order to obtain a result, $t \geq \frac{k+1}{k}\alpha + 6$ is required. Then the norm of the vector $\mathbf{q}$ fulfills the condition given by the Gaussian heuristic, that is, $||\mathbf{q}|| \leq \sqrt{\frac{d}{2\pi e}} \det(L)^{\frac{1}{d}}$, where d denotes the dimension of the lattice. We get the additive constant of 6 as the lattice dimension is smaller than the number of entries in the vector. Under the heuristic that a vector fulfilling this bound is indeed a shortest vector, $\mathbf{q}$ can be determined by lattice reduction. Jean-Charles Faugère et al. verify the heuristic experimentally.
Hence, the results obtained in the case of shared most significant bits are about the same as in the case of shared least significant bits. The lattices used are sublattices of $\mathbb{Z}^{\frac{(k+2)(k+1)}{2}}$ instead of $\mathbb{Z}^{k+1}$. This is due to the use of more combinations of the original equations. For an illustration compare Figure 4.3. Each combination of two composite numbers N_i and

$$\begin{pmatrix} 1 & N_0^{-1}N_1 & N_0^{-1}N_2 \\ 0 & 2^t & 0 \\ 0 & 0 & 2^t \end{pmatrix} \qquad \begin{pmatrix} 2^{n-t+\frac{1}{2}} & 0 & 0 & N_1 & N_2 & 0 \\ 0 & 2^{n-t+\frac{1}{2}} & 0 & -N_0 & 0 & N_2 \\ 0 & 0 & 2^{n-t+\frac{1}{2}} & 0 & -N_0 & -N_1 \end{pmatrix}$$

$\mathbf{B_L}$ in [MR09] $\mathbf{B_L}$ in [FMR09]

Figure 4.3: Comparison of the basis matrices $\mathbf{B_L}$ used in the constructions of [MR09] and [FMR09] in case of $k = 2$.

N_j can be used as it introduces a new monomial $(\tilde{p}_i - \tilde{p}_j)q_iq_j$. All the monomials can be described as linear combinations of q_i and q_j. In contrast to that, we introduce the new monomials $q_1, \ldots, q_k$ as (modular) linear functions of q_0. Hence, we only combine all N_i, $i = 1, \ldots, k$, with N_0. That is, we cannot add further equations. In the case of the lattice used by Jean-Charles Faugère et al., however, the additional combinations increase the size of the determinant of the lattice and the sublattice they use.
Note that the analysis of the determinant gets more complicated due to the rectangular shape of the basis matrix in the most significant bit case. For further details on the calculation of the bounds and the cases of more than two equations we refer the reader to the original work [FMR09].

4.4 Some General Criteria for Solving Systems of Multivariate Polynomial Equations via Shortest Vector Problems

In the previous sections some specific problems and their analyses via systems of equations have been presented. The problem of RSA with related messages can often be solved for any size of the unknowns by Groebner basis techniques (compare Section 4.1, for Groebner basis techniques consult Section 2.2). There is no need for lattice-based techniques in the analysis.
In what follows, we will only consider systems in which Groebner basis computations do not help to find the solutions. Two examples of such systems related to the problem of implicit factoring were given in Sections 4.2 and 4.3. They were solved by embedding the solutions into a lattice. Then the solutions were determined by solving a shortest vector problem. Based on these examples we derive criteria on which we can apply similar techniques to solve a system of equations by solving a shortest vector problem.

The analysis will be divided into two parts, the analysis of modular systems of equations and the analysis of systems of equations over the integers. Let us start with the modular case. Here the analyses of common and coprime moduli coincide so that we do not have to distinguish these two subcases. Thus, let us recall the general SMMPE problem presented in Definition 4.0.7. For given $k \in \mathbb{N}$, $\delta_1, \dots, \delta_k \in \mathbb{N}$, and $N_1, \dots, N_k \in \mathbb{N}$ the following system of multivariate polynomial equations shall be solved for its solutions $(\bar{x}_1, \dots, \bar{x}_l)$ such that $|\bar{x}_i| \le X_i \in \mathbb{R}$:

$$\begin{aligned} f_1(x_1,\dots,x_l) &\equiv 0 \pmod{N_1} \\ f_2(x_1,\dots,x_l) &\equiv 0 \pmod{N_2} \\ &\vdots \\ f_k(x_1,\dots,x_l) &\equiv 0 \pmod{N_k}\,. \end{aligned} \tag{4.13}$$

The polynomials $f_1(x_1,\dots,x_l) \in \mathbb{Z}_{N_1}[x_1,\dots,x_l], \dots, f_k(x_1,\dots,x_l) \in \mathbb{Z}_{N_k}[x_1,\dots,x_l]$ are of total degree $\delta_1, \dots, \delta_k$, respectively.
In order to solve the system of equations, we follow the approaches used in the previous sections. Thus, we would like to transform the system of equations into a system of the following structure:

$$\begin{aligned} \tilde{f}_1(x_1,\dots,x_l) := -c_1^{-1}\left(f_1(x_1,\dots,x_l) - c_1 m_1\right) &\equiv m_1 \pmod{N_1} \\ \tilde{f}_2(x_1,\dots,x_l) := -c_2^{-1}\left(f_2(x_1,\dots,x_l) - c_2 m_2\right) &\equiv m_2 \pmod{N_2} \\ &\vdots \\ \tilde{f}_k(x_1,\dots,x_l) := -c_k^{-1}\left(f_k(x_1,\dots,x_l) - c_k m_k\right) &\equiv m_k \pmod{N_k}\,. \end{aligned} \tag{4.14}$$

The parameter $m_i := m_i(x_1,\dots,x_l)$ denotes a specific monomial which occurs in the polynomial $f_i(x_1,\dots,x_l)$, the value c_i the corresponding coefficient. Without loss of generality we assume the coefficients c_i to be invertible modulo N_i. If they are not, we can determine divisors of the moduli N_i. Then by the Chinese Remainder Theorem 2.2.13 we get two new equations $f_{i1} \equiv 0 \pmod{N_{i1}}$ and $f_{i2} \equiv 0 \pmod{N_{i2}}$ equivalent to the old one. Then we modify the system by adding the system of the two new equations instead of the original equation. The new system is equivalent to the previous one.
In order to proceed analogously to the analysis in the examples, the monomials $m_1, \dots, m_k$ should not correspond to any row in the lattice, but only be introduced by linear combinations of these rows. For this property to hold, we require the set of monomials $\{m_1, \dots, m_k\}$ to be disjoint to the set of monomials $\mathcal{M} := \mathrm{Mon}(\{\tilde{f}_1, \dots, \tilde{f}_k\})$ which occur in the modified polynomials $\tilde{f}_i$. We enumerate the monomials of $\mathcal{M}$ as $m_{k+1}, \dots, m_{|\mathcal{M}|+k}$. Using this notation, we rewrite the equations of system (4.14) as

$$\sum_{j=1}^{|\mathcal{M}|} (\tilde{\mathbf{f}}_\mathbf{i})_j m_{k+j} \equiv m_i \pmod{N_i}\,, \quad i = 1, \dots, k\,.$$

The vector $\tilde{\mathbf{f}}_\mathbf{i}$ is the coefficient vector of $\tilde{f}_i$ with respect to $m_{k+1}, \dots, m_{|\mathcal{M}|+k}$. We define

$$L := \{(m_{k+1}, \dots, m_{|\mathcal{M}|+k}, m_1, \dots, m_k) \mid \sum_{j=1}^{|\mathcal{M}|} (\tilde{\mathbf{f}}_\mathbf{i})_j m_{k+j} \equiv m_i \pmod{N_i},\ i = 1, \dots, k\}\,.$$

The set L forms a discrete additive subgroup of $\mathbb{Z}^{|\mathcal{M}|+k}$. Thus, it defines a lattice. Note that in the examples of the previous sections we could just take the set $L = \{(x_1, \ldots, x_l) \mid f_i(x_1, \ldots, x_l) \equiv 0 \pmod{N_i}, \quad i = 1, \ldots, k\}$ as a lattice. This was possible as the equations were linear. In this more general case we have to include a linearization step by taking the monomials instead of the variables as components of the vectors. The lattice L is spanned by the row vectors of the basis matrix

$$\mathbf{B_L} := \begin{pmatrix} 1 & & & & & \\ & \ddots & & \tilde{\mathbf{f}}_{\mathbf{1}}^T & \ldots & \tilde{\mathbf{f}}_{\mathbf{k}}^T \\ & & 1 & & & \\ & & & N_1 & & \\ & & & & \ddots & \\ & & & & & N_k \end{pmatrix}.$$

To prove this, we proceed in the same manner as in Section 2.3. Let $\mathbf{B_L} = \begin{pmatrix} \mathbf{b_1} \\ \vdots \\ \mathbf{b}_{|\mathcal{M}|+\mathbf{k}} \end{pmatrix}$.
Then we have $\mathbf{b_j} \in L$ with $j = 1, \ldots, |\mathcal{M}|$ because $(\tilde{\mathbf{f}}_{\mathbf{i}})_j \cdot 1 \equiv (\tilde{\mathbf{f}}_{\mathbf{i}})_j \pmod{N_i}$ for $i = 1, \ldots, k$, and $\mathbf{b_j} \in L$ with $j = |\mathcal{M}| + 1, \ldots, |\mathcal{M}| + k$ because $0 \equiv N_i \pmod{N_i}$ and $0 \equiv 0 \pmod{N_l}$, $l \neq i$. Hence, $\langle \mathbf{B_L} \rangle \subseteq L$.
Now let $(m_{k+1}, \ldots, m_{|\mathcal{M}|+k}, m_1, \ldots, m_k) \in L$. Then the equation $\sum_{j=1}^{|\mathcal{M}|} (\tilde{\mathbf{f}}_{\mathbf{i}})_j m_{k+j} \equiv m_i \pmod{N_i}$ holds for $i = 1, \ldots, k$. Consequently, there exists an integer $n_i \in \mathbb{Z}$ such that $\sum_{j=1}^{|\mathcal{M}|} (\tilde{\mathbf{f}}_{\mathbf{i}})_j m_{k+j} + n_i N_i = m_i$. Then $(m_{k+1}, \ldots, m_{|\mathcal{M}|+k}, n_1, \ldots, n_k)\mathbf{B_L} = (m_{k+1}, \ldots, m_{|\mathcal{M}|+k}, m_1, \ldots, m_k) \in \langle \mathbf{B_L} \rangle$. Combining the two inclusions results in the claim $L = \langle \mathbf{B_L} \rangle$.

By construction we have $\bar{\mathbf{x}} := (m_{k+1}(\bar{x}_1, \ldots, \bar{x}_l), \ldots, m_{|\mathcal{M}|+k}(\bar{x}_1, \ldots, \bar{x}_l), m_1(\bar{x}_1, \ldots, \bar{x}_l), \ldots, m_k(\bar{x}_1, \ldots, \bar{x}_l)) \in L$. If $\bar{\mathbf{x}}$ was a shortest vector in L, we would be able to find it by lattice reduction. If, further, the solutions $\bar{x}_i$ could be written as rational functions of the monomials $m_1(\bar{x}_1, \ldots, \bar{x}_l), \ldots, m_{|\mathcal{M}|+k}(\bar{x}_1, \ldots, \bar{x}_l)$, then we would be able to determine the variables $\bar{x}_i$. A necessary condition for $\bar{\mathbf{x}}$ to be a shortest vector in L is that $||\bar{\mathbf{x}}|| \leq \sqrt{|\mathcal{M}| + k} \cdot \det(L)^{\frac{1}{|\mathcal{M}|+k}}$, i.e. $\bar{\mathbf{x}}$ has to be smaller than Minkowski's bound given in Definition 2.3.4.

Note that the vector $\bar{\mathbf{x}}$ is not balanced. Thus, its size is determined by its largest entry and it is rather unlikely to be a shortest vector in L. Hence, we change the vector to balance the sizes of its components. Therefore, we multiply its entries by suitable factors. Let $K := \operatorname{lcm}\{m_1(X_1, \ldots, X_l), \ldots, m_{|\mathcal{M}|+k}(X_1, \ldots, X_l)\}$, $K_i := \frac{K}{m_i(X_1, \ldots, X_l)}$ and $\bar{m}_i := m_i(\bar{x}_1, \ldots, \bar{x}_l)$. We set

$$\bar{\mathbf{y}} := (K_{k+1}\bar{m}_{k+1}, \ldots, K_{|\mathcal{M}|+k}\bar{m}_{|\mathcal{M}|+k}, K_1\bar{m}_1, \ldots, K_k\bar{m}_k).$$

Then every component of $\bar{\mathbf{y}}$ is approximately of size K. If we were able to determine the solutions using $\bar{\mathbf{x}}$, we would still be able to do so using $\bar{\mathbf{y}}$ as the K_i are known. However,

$\bar{\mathbf{y}} \notin L$. Thus, we define a modified lattice L_y via a basis matrix $\mathbf{B_{L_y}}$:

$$\mathbf{B_{L_y}} := \begin{pmatrix} K_{k+1} & & & & & \\ & \ddots & & K_1\tilde{\mathbf{f}}_{\mathbf{1}}^T & \dots & K_k\tilde{\mathbf{f}}_{\mathbf{k}}^T \\ & & K_{|\mathcal{M}|+k} & & & \\ & & & K_1N_1 & & \\ & & & & \ddots & \\ & & & & & K_kN_k \end{pmatrix}.$$

Let $l_i \in \mathbb{Z}$, $i = 1, \dots, k$, denote the integers such that $\sum_{j=1}^{|\mathcal{M}|}(\tilde{\mathbf{f}}_{\mathbf{i}})_j K_i m_{k+j} + l_i N_i = K_i m_i$. That is, $l_i = K_i n_i$. Then $\bar{\mathbf{y}} = (m_{k+1}(\bar{x}_1, \dots, \bar{x}_l), \dots, m_{|\mathcal{M}|+k}(\bar{x}_1, \dots, \bar{x}_l), l_1, \dots, l_k)\mathbf{B_{L_y}} \in L_y$. The vector $\bar{\mathbf{y}}$ has norm $||\bar{\mathbf{y}}|| \leq \sqrt{|\mathcal{M}| + k} \cdot K$. As we would like $\bar{\mathbf{y}}$ to be a shortest vector in $\mathbf{L_y}$, we get the necessary condition $||\bar{\mathbf{y}}|| \leq \sqrt{|\mathcal{M}| + k} \cdot \det(L_y)^{\frac{1}{|\mathcal{M}|+k}}$ by Minkowski's bound. Remark that due to the adaptation of the matrix $\det(L_y) \geq \det(L)$. It is $\det(L_y) = \prod_{j=1}^{|\mathcal{M}|+k} K_j \prod_{i=1}^{k} N_i$. Thus, we have the condition:

$$\begin{aligned} \sqrt{|\mathcal{M}| + k} \left(\prod_{j=1}^{|\mathcal{M}|+k} K_j \prod_{i=1}^{k} N_i \right)^{\frac{1}{|\mathcal{M}|+k}} &\geq \sqrt{|\mathcal{M}| + k} \cdot K \\ \Leftrightarrow \quad \prod_{i=1}^{k} N_i &\geq \prod_{j=1}^{|\mathcal{M}|+k} \underbrace{\frac{K}{K_j}}_{=m_j(X_1,\dots,X_l)}. \end{aligned}$$

Based on this, we make the following heuristic assumption.

Assumption 4.4.1
Let N_i, $i = 1, \dots, k$, and m_j, $j = 1, \dots, |\mathcal{M}| + k$, be defined as in the system of equations (4.14). Further, let $\mathbf{b_1}$ be a shortest vector in L_y. Then $\mathbf{b_1} = \pm\bar{\mathbf{y}}$.

For a complete analysis we add another assumption.

Assumption 4.4.2
Let $m_1, \dots, m_{|\mathcal{M}|+k}$ be defined with respect to the system of equations (4.14). Then the values $\bar{x}_1, \dots, \bar{x}_l$ can be uniquely determined using $m_1(\bar{x}_1, \dots, \bar{x}_l), \dots, m_{|\mathcal{M}|+k}(\bar{x}_1, \dots, \bar{x}_l)$.

The results of the method described above can be summarized in the following theorem.

Theorem 4.4.3
Let the system of equations (4.13) be given. Further, let m_j, $j = 1, \dots, |\mathcal{M}|+k$, be defined as in the system of equations (4.14).
Then on Assumptions 4.4.1 and 4.4.2 we can determine all solutions $|\bar{x}_i| \leq X_i$, $i = 1, \dots, k$, of system (4.13) in time polynomial in $((|\mathcal{M}| + k)^{\frac{|\mathcal{M}|+k}{2}}, \max_i\{\log(KN_i)\})$ if

$$\prod_{i=1}^{k} N_i \geq \prod_{j=1}^{|\mathcal{M}|+k} m_j(X_1, \dots, X_l).$$

The running time is dominated by the computation of a shortest vector [Kan87, Hel85]. Remark that the different equations which are used as input of the theorem have to be independent of one another. If, for example, the same equation is added twice in the lattice, the heuristic fails.

The SIMPE-problem can be analyzed in an analogous manner. However, some additional problems may occur. We give a sketch of the method to point out the problems and additional conditions we require such that this approach still works. First, the system of equations

$$\begin{array}{rcl} f_1(x_1,\ldots,x_l) & = & 0 \\ f_2(x_1,\ldots,x_l) & = & 0 \\ & \vdots & \\ f_k(x_1,\ldots,x_l) & = & 0 \end{array} \tag{4.15}$$

is transformed into a system of equations

$$\begin{array}{rcl} \tilde{f}_1(x_1,\ldots,x_l) := -\left(f_1(x_1,\ldots,x_l) - c_1m_1\right) & = & c_1m_1 \\ \tilde{f}_2(x_1,\ldots,x_l) := -\left(f_2(x_1,\ldots,x_l) - c_2m_2\right) & = & c_2m_2 \\ & \vdots & \\ \tilde{f}_k(x_1,\ldots,x_l) := -\left(f_k(x_1,\ldots,x_l) - c_km_k\right) & = & c_km_k\,. \end{array} \tag{4.16}$$

As all calculations are performed over the integers, we can no longer assume that the coefficients c_i are invertible. Hence, a natural adaptation would be to directly construct a lattice L_I using the equations of system (4.16). Analogously to the modular case, the solutions of system (4.16) (taking c_im_i instead of m_i) form a lattice L_I for which we can give a basis $\mathbf{B_{L_I}}$:

$$\mathbf{B_{L_I}} := \begin{pmatrix} K_{k+1} & & & & & \\ & \ddots & & K_1\tilde{\mathbf{f}}_{\mathbf{1}}^T & \ldots & K_k\tilde{\mathbf{f}}_{\mathbf{k}}^T \\ & & K_{|\mathcal{M}|+k} & & & \end{pmatrix}.$$

Remark that in this construction the values $c_1m_1,\ldots,c_km_k$ correspond to entries of our target vector. As we intend to determine the target vector by calculating a shortest vector in the lattice L_I, we need these values to be small. Thus, for the method to work, we have to make an additional claim: The coefficients c_i have to be of small constant size.
Furthermore, note that the lattice L_I is not of full dimension. Any basis of this lattice is, thus, given by a rectangular matrix. For rectangular matrices determinant calculations are more complicated than for square upper triangular matrices. Consequently, it is more difficult to calculate bounds on the solutions in the integer case. Therefore, we can no longer give a general bound here. It has to be calculated individually with respect to the specific system of equations.

In general, the method presented in this chapter works as follows. First, we have to transform the given system of equations into a lattice in which our solution corresponds to a shortest vector $\mathbf{v}$. Then we can determine the vector $\mathbf{v}$ by lattice reduction. The basic condition on which the target vector $\mathbf{v}$ is a shortest vector is given by Assumption 4.4.1. This, however, is a rather strong requirement. Therefore, only very small solutions, if any, can be found. It would be advantageous if we could relax this condition. A method in which the target vector no longer has to be a shortest vector, but only smaller than some basis vector has been given by Don Coppersmith in 1996 [Cop96b, Cop96a]. For some problems this method allows to find larger solutions than the method presented in this chapter. We will use variants of it to solve systems of equations in the following chapters.

Chapter 5

Solving Systems of Multivariate Polynomial Equations with Coppersmith's Method

In 1996, Don Coppersmith presented a lattice-based technique to solve multivariate polynomial equations over the integers as well as modular ones. A description of the technique, which we call Coppersmith's method, is given in Section 2.3. The original work mainly refers to bivariate equations. Ellen Jochemsz and Alexander May show how to construct shift polynomial sets for multivariate equations in more variables in [JM06]. Directly applying Coppersmith's method with those shift polynomial sets, we can give upper bounds on the unknowns in case of one equation. In the following sections we will make some progress in generalizing these techniques to systems of multivariate equations.

5.1 Coppersmith's Algorithm with Systems of Equations

Let us briefly recall how Coppersmith's method to find small solutions of multivariate equations works in case of one equation: First, a set of shift polynomials is chosen and a lattice is defined using these shift polynomials. In a second step, a sublattice of vectors having zeros in the last components is determined. This sublattice is constructed to have the same determinant as the original lattice. In order for this to hold, there has to exist a unimodular transformation $\mathbf{U}$ such that $\mathbf{UF} = \begin{pmatrix} \mathbf{0} \\ \mathbf{I} \end{pmatrix}$, where the matrix $\mathbf{F}$ is induced by a given set $\mathcal{F}$ of shift polynomials. By Theorem 2.1.3 such a transformation $\mathbf{U}$ exists if the elementary divisors of $\mathbf{F}$ are all equal to one. In case of $k = 1$ (i. e. the system consists of a single equation), which is described in the original work of Don Coppersmith [Cop96b], the condition can be verified easily. Let f be the polynomial the roots of which are to be determined. As f is monic, the vector corresponding to this polynomial contains an entry equal to 1. Consequently, each shift polynomial is monic as well as it is constructed by multiplying f with some monomial. In the shift set, the shifts can be ordered by graded

lexicographic ordering so that each polynomial introduces at least one new monomial, namely its leading monomial. It is at a position in which all previous vectors were of value zero. Thus, the matrix $\mathbf{F}$ corresponding to the set of shift polynomials contains an upper triangular submatrix with a diagonal of value one. Consequently, all elementary divisors of $\mathbf{F}$ are equal to one and the required unimodular transformation $\mathbf{U}$ exists.
The same holds for the analysis of a single multivariate polynomial equation using the shift polynomial set $\mathcal{F}$ as defined in [JM06]. There the polynomial to be analyzed is transformed into a monic one in a first step. Then the shift polynomial set is calculated to consist of shifted powers of the original polynomial. These polynomials are ordered in a way that each shift polynomial introduces its leading monomial. Thus, the matrix $\mathbf{F}$ again contains an upper triangular submatrix having only ones on its diagonal. This implies that all elementary divisors of $\mathbf{F}$ are equal to one and a suitable unimodular transformation $\mathbf{U}$ exists.

For a general set of shift polynomials, however, this is not that easy to see.
In what follows we will relate the condition of Theorem 2.1.3 to conditions on the set of shift polynomials. Therefore, we introduce some additional notation.
When constructing sets of shift polynomials, the goal is to include additional information with every additional polynomial. If a polynomial is an integer linear combination of the polynomials already included, we do not gain anything by adding it to the shift polynomial set. Hence, the polynomials are constructed to be linearly independent. This is a necessary condition on the existence of a suitable transformation of the lattice. To see why, assume $\mathbf{F}$ to be the matrix induced by a linearly dependent set of shift polynomials $\mathcal{F}$. Then there is a non-trivial linear combination of polynomials $f \in \mathcal{F}$ which adds up to the zero polynomial. The same non-trivial linear combination of the corresponding vectors $\mathbf{f}^T$ adds up to the zero vector. Thus, at least one elementary divisor of $\mathbf{F}$ is equal to 0 and the required unimodular transformation of $\mathbf{F}$ does not exist.
One might assume that linear independence immediately implies that the construction of the sublattice with the same determinant works. However, this is not the case as can be seen in the following example.

Example 5.1.1
Let $N \in \mathbb{Z}\backslash\{-1,0,1\}$ and $f(x_1,x_2) = x_1x_2+Nx_1$ and $g(x_1,x_2) = x_1x_2+Nx_2 \in \mathbb{Z}[x_1,x_2]$ be two bivariate polynomials over the integers. Both polynomials contain different monomials. Thus, they are linearly independent over the integers. Building a Coppersmith type basis matrix with the shift polynomial set $\mathcal{F} := \{f, g\}$, we get

$$\mathbf{B} := \begin{array}{c} \left(\begin{array}{ccc} (X_1X_2)^{-1} & 1 & 1 \\ 0 & N & 0 \\ 0 & 0 & N \end{array} \right) \end{array} \begin{array}{l} x_1x_2 \\ x_1 \\ x_2 \end{array} .$$

In order to check whether we can determine a sublattice with the same determinant, we only regard the submatrix $\mathbf{F}$ corresponding to the shift polynomial set. That is, we regard the matrix consisting of the last two columns. First, we apply unimodular transformations *in order to transform the first column. We permute the first and second row and then eliminate the entry N in the first column. By this we obtain*

$$\tilde{\mathbf{U}}\mathbf{F} := \begin{pmatrix} 0 & -N \\ 1 & 1 \\ 0 & N \end{pmatrix} .$$

Having a look at the matrix $\tilde{\mathbf{U}}\mathbf{F}$, we can directly see that the elementary divisors are 1 and N. By Theorem 2.1.3, this implies that we do not get a unimodular transformation $\mathbf{U}$ such that $\mathbf{UF} = \begin{pmatrix} \mathbf{0} \\ \mathbf{I} \end{pmatrix}$. Instead of a second elementary divisor of value 1, we obtained one of value N. Looking at $f(x_1, x_2)$ and $g(x_1, x_2)$ modulo N, we observe that $f(x_1, x_2) \equiv x_1 x_2 \equiv g(x_1, x_2) \pmod{N}$, i. e. the two polynomials are linearly dependent, here even equal, in $\mathbb{Z}_N[x_1, x_2]$.

The observation made in the above example can be generalized easily. Instead of using polynomials which are linearly independent over the integers, we only use polynomials which are linearly independent over the integers as well as in any ring $\mathbb{Z}_a$ with $1 \neq a \in \mathbb{N}$. In contrast to the previously used condition, this one is not only an intuition, but provably equivalent to the existence of a sublattice with the same determinant containing our target vector. In the subsequent of this section, we will give conditions on which we can restrict the number of potential moduli. At the moment, however, we use the general statement. Note that modularly dependent polynomials could still be used in the lattice construction. They do not cause the construction to fail, but do no longer allow for a direct computation of the determinant. That is, a priori computations give too large upper bounds. In practice, the determinant of the sublattice then is significantly smaller. In many examples, this implies that we do not obtain interesting bounds on possible solutions any more.
Let $l \in \mathbb{N}$ and $\mathcal{F}$ be a set of polynomials in $\mathbb{Z}[x_1, \ldots, x_l]$. Without loss of generality we assume $\mathcal{F}$ to be ordered and denote the i-th polynomial in $\mathcal{F}$ by f_i. Let $w := |\mathrm{Mon}(\mathcal{F})|$ and $m_1, \ldots, m_w$ be an ordering of all monomials of $\mathrm{Mon}(\mathcal{F})$. Let $\mathbf{f}$ be the vector of the coefficients of f such that $(\mathbf{f})_i$ is the coefficient of the monomial m_i. Let $\mathbf{F}$ be the matrix whose column vectors are $\mathbf{f}^T$ for all $f \in \mathcal{F}$.

Definition 5.1.2
*The set of polynomials $\mathcal{F} \subset \mathbb{Z}[x_1, \ldots, x_l]$ is called a **determinant preserving set** if and only if there exists a unimodular transformation $\mathbf{U}$ such that*

$$\mathbf{UF} = \begin{pmatrix} \mathbf{0}^{(\mathbf{w}-|\mathcal{F}|)\times|\mathcal{F}|} \\ \mathbf{I}_{|\mathcal{F}|} \end{pmatrix} .$$

Note that any shift polynomial set $\mathcal{G}$ giving a good bound although it is not determinant preserving corresponds to a smaller determinant preserving set $\mathcal{F}$ with the same bound. The set $\mathcal{F}$ can be constructed by appropriately removing polynomials from $\mathcal{G}$. Therefore, we can restrict the analysis directly to determinant preserving shift polynomial sets.

Remark 5.1.3
In the introduction of this section we have already shown that the shift polynomial sets Don Coppersmith [Cop96b] and Ellen Jochemsz and Alexander May [JM06] derive for one (multivariate) polynomial are determinant preserving.

In Example 5.1.1 the elementary divisors of $\mathbf{F}$ are 1 and N. Therefore, we know that no appropriate unimodular transformation exists. The set $\{f, g\}$ is, thus, not determinant preserving.
However, changing the example slightly by setting $f_2(x_1, x_2) = x_1x_2 + Nx_1$ and $g_2(x_1, x_2) = x_1x_2 + Mx_2$ with $\gcd(M, N) = 1$, the situation changes. The last two columns of the basis matrix become

$$\mathbf{F_2} := \begin{array}{c}\left(\begin{array}{cc} 1 & 1 \\ N & 0 \\ 0 & M \end{array}\right)\end{array}\begin{array}{l} x_1x_2 \\ x_1 \\ x_2 \end{array} .$$

Applying the same unimodular transformations, we get

$$\tilde{\mathbf{U}}\mathbf{F_2} := \left(\begin{array}{cc} 0 & -N \\ 1 & 1 \\ 0 & M \end{array}\right) .$$

As the greatest common divisor of M and N is 1, this can easily be transformed to

$$\mathbf{U}\tilde{\mathbf{U}}\mathbf{F_2} := \left(\begin{array}{cc} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{array}\right) .$$

Consequently, the set $\{f_2, g_2\}$ is determinant preserving. Compared to f and g, the new polynomials are not linearly dependent modulo N or M or any other integer greater than one either.
We generalize this observation to obtain a criterion for determinant preserving sets.

Theorem 5.1.4
Let $\mathcal{F}$ be a determinant preserving set and $\mathcal{F} \cup \{g\} \subset \mathbb{Z}[x_1, \dots, x_l]$ linearly independent over the integers.
Then $\mathcal{G} := \mathcal{F} \cup \{g\}$ is a determinant preserving set if and only if for all $1 \neq a \in \mathbb{N}$ the polynomial g is linearly independent of $\mathcal{F}$ in $\mathbb{Z}_a$.

PROOF: Let $\mathcal{F} = \{f_1, \dots, f_{|\mathcal{F}|}\}$ be an enumeration of the shift polynomials.
We prove this theorem by proving the contraposition:
"The set $\mathcal{G} = \mathcal{F} \cup \{g\}$ is not determinant preserving if and only if there is an $1 \neq a \in \mathbb{N}$ such that $g \equiv \sum_{j=1}^{|\mathcal{F}|} c_j f_j \pmod{a}$ with $c_j \in \mathbb{Z}$."
First, let us introduce some additional values. Let $w_{\mathcal{F}} := |\mathrm{Mon}(\mathcal{F})|$, $w_g := |\mathrm{Mon}(g) \setminus \mathrm{Mon}(\mathcal{F})|$, and $m_{w_{\mathcal{F}}+1}, \dots, m_{w_{\mathcal{F}}+w_g}$ be an enumeration of the monomials occurring in g, but not in $\mathcal{F}$. Then let $w := w_{\mathcal{F}} + w_g$ denote the number of all monomials occurring in $\mathcal{G}$. Note that the case of $w_g = 0$ is also included in the following proof although the proof could be simplified in this special case.

We start by proving "If there is $1 \neq a \in \mathbb{N}$ such that $g \equiv \sum_{j=1}^{|\mathcal{F}|} c_j f_j \pmod{a}$ with $c_j \in \mathbb{Z}$, then $\mathcal{G} = \mathcal{F} \cup \{g\}$ is not determinant preserving".

Suppose there are $1 \neq a \in \mathbb{N}$ and $c_j \in \mathbb{Z}_a$ such that $g \equiv \sum_{j=1}^{|\mathcal{F}|} c_j f_j \pmod{a}$. Then it holds that $\mathbf{g}^T \equiv \sum_{j=1}^{|\mathcal{F}|} c_j \mathbf{f_j}^T \pmod{a}$. Here we define the vectors with respect to the ordering $m_1, \ldots, m_w$. This means, the last w_g components of vectors corresponding to elements of $\mathcal{F}$ are zero modulo a.
From the preconditions we know that there exists $\mathbf{V}$ such that $\mathbf{VF} = \begin{pmatrix} \mathbf{0}^{(\mathbf{w}_{\mathcal{F}}-|\mathcal{F}|)\times|\mathcal{F}|} \\ \mathbf{I}_{|\mathcal{F}|} \end{pmatrix}$.
Here, the columns of $\mathbf{F}$ are only taken with regard to the monomials $m_1, \ldots, m_{w_{\mathcal{F}}}$. We extend this with regard to $m_1, \ldots, m_w$ by adding w_g zero rows to $\mathbf{F}$ and extending $\mathbf{V}$ as

$$\mathbf{U} := \begin{pmatrix} \mathbf{V} & \mathbf{0}^{\mathbf{w}_{\mathcal{F}}\times\mathbf{w_g}} \\ \mathbf{0}^{\mathbf{w_g}\times\mathbf{w}_{\mathcal{F}}} & \mathbf{I_{w_g}} \end{pmatrix}.$$

For the ease of notation, we denote the extended version of $\mathbf{F}$ as well by $\mathbf{F}$. Let $\mathbf{G}$ denote the matrix constructed by taking the matrix $\mathbf{F}$ and adding the vector $\mathbf{g}^T$ as last column. If not stated otherwise, we are referring to vectors with regard to all the monomials $m_1, \ldots, m_w$. Then $\mathbf{Ug}^T \equiv \sum_{j=1}^{|\mathcal{F}|} c_j \mathbf{Uf_j}^T \pmod{a} \equiv \sum_{j=1}^{|\mathcal{F}|} c_j (\mathbf{e}^{\mathbf{w}_{\mathcal{F}}-|\mathcal{F}|+\mathbf{j}})^T \pmod{a}$ with $(\mathbf{e}^{\mathbf{w}_{\mathcal{F}}-|\mathcal{F}|+\mathbf{j}})^T$ being the $(w_{\mathcal{F}} - |\mathcal{F}| + j)$-th unit vector of length w. Consequently,

$$\mathbf{Ug}^T = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ c_1 \\ \vdots \\ c_{|\mathcal{F}|} \\ 0 \\ \vdots \\ 0 \end{pmatrix} \pmod{a}$$

and with appropriate $z_i \in \mathbb{Z}$

$$\mathbf{UG} = \begin{pmatrix} & az_1 \\ \mathbf{0}^{(\mathbf{w}_{\mathcal{F}}-|\mathcal{F}|)\times|\mathcal{F}|} & \vdots \\ & az_{(w_{\mathcal{F}}-|\mathcal{F}|)} \\ & c_1 + az_{(w_{\mathcal{F}}-|\mathcal{F}|+1)} \\ \mathbf{I}_{|\mathcal{F}|} & \vdots \\ & c_{|\mathcal{F}|} + az_{w_{\mathcal{F}}} \\ & az_{(w_{\mathcal{F}}+1)} \\ \mathbf{0}^{\mathbf{w_g}\times|\mathcal{F}|} & \vdots \\ & az_w \end{pmatrix}.$$

Eliminating the entries c_1 to $c_{|\mathcal{F}|}$ by column operations one can directly see that a divides all remaining values in the last column and, thus, the last elementary divisor of $\mathbf{G}$. Applying Theorem 2.1.3, we derive that $\mathcal{G}$ is not determinant preserving.

Now we prove "If the set $\mathcal{G} = \mathcal{F} \cup \{g\}$ is not determinant preserving, then there is an $1 \neq a \in \mathbb{Z}$ such that $g \equiv \sum_{j=1}^{|\mathcal{F}|} c_j f_j \pmod{a}$ with $c_j \in \mathbb{Z}$".
Suppose $\mathcal{G} = \mathcal{F} \cup \{g\}$ is not determinant preserving, that is, by Theorem 2.1.3, we either have that the last elementary divisor of $\mathbf{G}$ is not equal to 1 or the number of elementary divisors of $\mathbf{G}$ is smaller than $|\mathcal{G}|$.
First, we assume that at least one elementary divisor is equal to $a \neq 1$. Our goal is to show that there exist coefficients $c_1, \ldots, c_{|\mathcal{F}|} \in \mathbb{Z}_a$ such that $g \equiv \sum_{j=1}^{|\mathcal{F}|} c_j f_j \pmod{a}$. Therefore, we start by setting $g \equiv \sum_{j=1}^{|\mathcal{F}|} \tilde{c}_j f_j + \tilde{r} \pmod{a}$ (such a representation always exists, take for example $\tilde{r} = g$ and $\tilde{c}_j = 0$). The aim is to change the coefficients $\tilde{c}_j$ until we have $\tilde{r} \equiv 0 \pmod{a}$.

Let $\mathbf{U}$ again be the unimodular transformation such that $\mathbf{UF} = \begin{pmatrix} \mathbf{0}^{(\mathbf{w}_{\mathcal{F}}-|\mathcal{F}|)\times|\mathcal{F}|} \\ \mathbf{I}_{|\mathcal{F}|} \\ \mathbf{0}^{\mathbf{w_g}\times|\mathcal{F}|} \end{pmatrix}$.

Then

$$\begin{aligned}
\mathbf{Ug}^T &\equiv \sum_{j=1}^{|\mathcal{F}|} \tilde{c}_j \mathbf{U}\mathbf{f_j}^T + \mathbf{U}\tilde{\mathbf{r}}^T \pmod{a} \\
&\equiv \sum_{j=1}^{|\mathcal{F}|} \tilde{c}_j (\mathbf{e}^{\mathbf{w}_{\mathcal{F}}-|\mathcal{F}|+\mathbf{j}})^T + \mathbf{U}\tilde{\mathbf{r}}^T \pmod{a} \\
&\equiv \begin{pmatrix} (\mathbf{U}\tilde{\mathbf{r}}^T)_1 \\ \vdots \\ (\mathbf{U}\tilde{\mathbf{r}}^T)_{w_{\mathcal{F}}-|\mathcal{F}|} \\ \tilde{c}_1 + (\mathbf{U}\tilde{\mathbf{r}}^T)_{w_{\mathcal{F}}-|\mathcal{F}|+1} \\ \vdots \\ \tilde{c}_{|\mathcal{F}|} + (\mathbf{U}\tilde{\mathbf{r}}^T)_{w_{\mathcal{F}}} \\ (\mathbf{U}\tilde{\mathbf{r}}^T)_{w_{\mathcal{F}}+1} \\ \vdots \\ (\mathbf{U}\tilde{\mathbf{r}}^T)_w \end{pmatrix} \pmod{a}.
\end{aligned}$$

By this we get $\gcd((\mathbf{U}\tilde{\mathbf{r}}^T)_1, \ldots, (\mathbf{U}\tilde{\mathbf{r}}^T)_{w_{\mathcal{F}}-|\mathcal{F}|}, (\mathbf{U}\tilde{\mathbf{r}}^T)_{w_{\mathcal{F}}+1}, \ldots, (\mathbf{U}\tilde{\mathbf{r}}^T)_w) = a > 1$.
If $\gcd((\mathbf{U}\tilde{\mathbf{r}}^T)_1, \ldots, (\mathbf{U}\tilde{\mathbf{r}}^T)_{w_{\mathcal{F}}-|\mathcal{F}|}, (\mathbf{U}\tilde{\mathbf{r}}^T)_{w_{\mathcal{F}}+1}, \ldots, (\mathbf{U}\tilde{\mathbf{r}}^T)_w) = b \neq a$ held, we could perform the elementary divisor algorithm on $\mathbf{UG}$, eliminate the j-th entries for $j = w_{\mathcal{F}} - |\mathcal{F}| + 1, \ldots, w_{\mathcal{F}}$ by column operations, then set the next diagonal entry to b and eliminate all further entries in this column by row operations. This implies that the last elementary divisor is equal to b which contradicts the preconditions. Thus, $(\mathbf{U}\tilde{\mathbf{r}}^T)_j \equiv 0 \pmod{a}$ for $j = 1, \ldots, w_{\mathcal{F}} - |\mathcal{F}|, w_{\mathcal{F}} + 1, \ldots, w$.

Further, we need that $(\mathbf{U}\tilde{\mathbf{r}}^T)_j \equiv 0 \pmod a$ for $j = w_{\mathcal{F}} - |\mathcal{F}| + 1, \ldots, w_{\mathcal{F}}$. Assume that this does not hold, i. e. we have $(\mathbf{U}\tilde{\mathbf{r}}^T)_{w_{\mathcal{F}}-|\mathcal{F}|+l} \equiv d \pmod a$ for an index $l \in \{1, \ldots, |\mathcal{F}|\}$. Then define $c_l := \tilde{c}_l + d$ and $r := \tilde{r} - df_l$ and $c_j := \tilde{c}_j$ for all $j \neq l$.
This implies $\sum_{j=1}^{|\mathcal{F}|} c_j f_j + r \pmod a \equiv \sum_{j=1, j\neq l}^{|\mathcal{F}|} \tilde{c}_j f_j + \tilde{c}_l f_l + df_l + \tilde{r} - df_l \equiv g \pmod a$.
Following the same line of argumentation used before, we have

$$\begin{aligned}
\mathbf{U}\mathbf{g}^T &\equiv \sum_{j=1}^{|\mathcal{F}|} c_j \mathbf{U}\mathbf{f_j}^T + \mathbf{U}\mathbf{r}^T \pmod a \\
&\equiv \sum_{j=1}^{|\mathcal{F}|} c_j (\mathbf{e}^{\mathbf{w}-|\mathcal{F}|+\mathbf{j}})^T + \mathbf{U}\mathbf{r}^T \pmod a \\
&\equiv \begin{pmatrix} (\mathbf{U}\mathbf{r}^T)_1 \\ \vdots \\ (\mathbf{U}\mathbf{r}^T)_{w_{\mathcal{F}}-|\mathcal{F}|} \\ c_1 + (\mathbf{U}\mathbf{r}^T)_{w_{\mathcal{F}}-|\mathcal{F}|+1} \\ \vdots \\ c_{|\mathcal{F}|} + (\mathbf{U}\mathbf{r}^T)_{w_{\mathcal{F}}} \\ (\mathbf{U}\mathbf{r}^T)_{w_{\mathcal{F}}+1} \\ \vdots \\ (\mathbf{U}\mathbf{r}^T)_w \end{pmatrix} \pmod a
\end{aligned}$$

with $(\mathbf{U}\mathbf{r}^T)_j \equiv 0 \pmod a$ for $j = 1, \ldots, w_{\mathcal{F}} - |\mathcal{F}|, w_{\mathcal{F}} + 1, \ldots, w$. Moreover, we have $(\mathbf{U}\mathbf{r}^T)_{w_{\mathcal{F}}-|\mathcal{F}|+l} \equiv (\mathbf{U}\tilde{\mathbf{r}}^T)_{w_{\mathcal{F}}-|\mathcal{F}|+l} - \mathbf{U}(d\mathbf{f_l}^T)_{w_{\mathcal{F}}-|\mathcal{F}|+l} \equiv d - (d(\mathbf{e}^{\mathbf{w}-|\mathcal{F}|+\mathbf{l}})^T)_{w_{\mathcal{F}}-|\mathcal{F}|+l} \equiv d - d \equiv 0 \pmod a$. Performing this translation of c_l for all $l \in \{1, \ldots, |\mathcal{F}|\}$ such that $(\mathbf{U}\mathbf{r})^{\mathbf{T}}_{\mathbf{w}_{\mathcal{F}}-|\mathcal{F}|+\mathbf{l}} \not\equiv 0 \pmod a$, we get $\mathbf{U}\mathbf{r}^T \equiv \mathbf{0}^{\mathbf{w}\times\mathbf{1}} \pmod a$ and, as $\mathbf{U}$ is unimodular and, thus, invertible, we get $\mathbf{r}^T \equiv \mathbf{0}^{\mathbf{w}\times\mathbf{1}} \pmod a$. Therefore, $g \equiv \sum_{j=1}^{|\mathcal{F}|} c_j f_j \pmod a$, which concludes the proof.

In case of $\mathbf{G}$ having only $|\mathcal{F}|$ elementary divisors, we know that there exist unimodular matrices $\mathbf{U}$ and $\mathbf{V}$ such that $\mathbf{UGV} = \mathrm{Diag}(a_1, \ldots, a_{|\mathcal{F}|}, 0)$. Thus, $\mathrm{rank}(\mathbf{G}) = \mathrm{rank}(\mathbf{UGV}) < |\mathcal{G}| = |\mathcal{F}| + 1$. This implies that the columns of $\mathbf{G}$ are linearly dependent over $\mathbb{Z}$, that is, there exist $c_1, \ldots, c_{|\mathcal{F}|}, c_{|\mathcal{F}|+1} \in \mathbb{Z}$ such that $\sum_{i=1}^{|\mathcal{F}|+1} c_i \mathbf{G}_{\cdot,i} = \mathbf{0}^{\mathbf{w}\times\mathbf{1}}$. This implies $\sum_{i=1}^{|\mathcal{F}|} c_i f_i + c_{|\mathcal{F}|+1} g = 0$, i. e. $\mathcal{G}$ is linearly dependent. This contradicts the preconditions. Thus, the theorem is proven. ■

As an extension of the theorem we get the following result.

Theorem 5.1.5
Let $\mathcal{F} \subset \mathbb{Z}[x_1, \ldots, x_l]$ be a set of polynomials.
Then $\mathcal{F}$ is determinant preserving if and only if for all $f \in \mathcal{F}$ and all $1 \neq a \in \mathbb{N}$ the polynomial f is linearly independent of $\mathcal{F} \setminus \{f\}$ in $\mathbb{Z}_a[x_1, \ldots, x_l]$.

PROOF: As a direct application of Theorem 5.1.4 setting $g = f$ we have "If there is a polynomial $f \in \mathcal{F}$ and a parameter $1 \neq a \in \mathbb{N}$ such that f is linearly dependent of

$\mathcal{F} \setminus \{f\}$, then $\mathcal{F}$ is not determinant preserving". This is equivalent to "If $\mathcal{F}$ is determinant preserving, then for all $f \in \mathcal{F}$ and all $1 \neq a \in \mathbb{N}$ the polynomial f is linearly independent of $\mathcal{F} \setminus \{f\}$ in $\mathbb{Z}_a[x_1, \ldots, x_l]$".
We show the opposite direction of the equivalence "If for all $f \in \mathcal{F}$ and all $1 \neq a \in \mathbb{N}$ the polynomial f is linearly independent of $\mathcal{F} \setminus \{f\}$ in $\mathbb{Z}_a[x_1, \ldots, x_l]$, then $\mathcal{F}$ is determinant preserving" by induction on the size of $\mathcal{F}$ using Theorem 5.1.4 in the inductive step.
First assume $|\mathcal{F}| = 1$. Then $\mathcal{F} = \{f\}$ for a non-zero polynomial f. Furthermore, the positive greatest common divisor of the coefficients of f is equal to 1. In any other case, if $\gcd(\mathbf{f}) = a > 1$, then $f \equiv 0 \pmod{a}$, i.e. we have linear dependence in $\mathbb{Z}_a$ which contradicts the precondition. The claim then follows by Lemma 2.1.4, i.e. we have that $\{f\}$ is determinant preserving.
The **hypothesis** we use is as follows:
For an $n \in \mathbb{N}$ and any set of polynomials $\mathcal{F}$ such that $|\mathcal{F}| = n$ it holds that if for all $f \in \mathcal{F}$ and all $1 \neq a \in \mathbb{N}$ the polynomial f is linearly independent of $\mathcal{F} \setminus \{f\}$ in $\mathbb{Z}_a$, then $\mathcal{F}$ is determinant preserving.
We continue with the inductive step. Let $\mathcal{F}$ be a set of polynomials fulfilling the precondition with $|\mathcal{F}| = n + 1$. Choose an arbitrary $g \in \mathcal{F}$. Then $|\mathcal{F} \setminus \{g\}| = n$, and by the precondition it holds for all $f \in \mathcal{F} \setminus \{g\}$ and all $1 \neq a \in \mathbb{N}$ that the polynomial f is linearly independent of $\mathcal{F} \setminus \{f, g\}$ in $\mathbb{Z}_a$. Therefore, $\mathcal{F} \setminus \{g\}$ is determinant preserving by the hypothesis.
From the preconditions we further get that for all $1 \neq a \in \mathbb{N}$ the polynomial g is linearly independent of $\mathcal{F} \setminus \{g\}$ in $\mathbb{Z}_a[x_1, \ldots, x_l]$. In order to apply Theorem 5.1.4 we have to verify the second precondition that the set $\mathcal{F}$ is linearly independent over the integers. Assume on the contrary that there exist integers $c_j \in \mathbb{Z}$ such that $\sum_{j=1}^{n+1} c_j f_j = 0$. Without loss of generality let $g = f_{n+1}$. Then $c_{n+1} \notin \{-1, 0, 1\}$. If $c_{n+1} = 0$, then $\mathcal{F} \setminus \{g\}$ is not determinant preserving. If $c_{n+1} = \pm 1$, then it is $g = \sum_{j=1}^{n} (\pm c_j) f_j$ over the integers and, consequently also modulo a for any $a \in \mathbb{N}$. Both implications contradict the preconditions. Thus, it is $\sum_{j=1}^{n} c_j f_j \equiv 0 \pmod{c_{n+1}}$. Without loss of generality we have $\bar{i} \in \{1, \ldots, n\}$ such that $f_{\bar{i}} \in \mathcal{F} \setminus \{g\}$ and $\gcd(c_{\bar{i}}, c_{n+1}) = 1$. (If not, either all c_j share a common divisor or different c_j have different common divisors with c_{n+1}. In the first case we can divide the whole equation by this divisor, in the second case we can just take $\gcd(c_j, c_{n+1})$ for some arbitrary j as new modulus.)
Therefore, we can rewrite the equation as $f_{\bar{i}} \equiv \sum_{j=1, j \neq \bar{i}}^{n} (-c_{\bar{i}}^{-1}) c_j f_j \pmod{c_{n+1}}$, i.e. $f_{\bar{i}}$ is not linearly independent of $\mathcal{F} \setminus \{g, f_{\bar{i}}\}$ in $\mathbb{Z}_{c_{n+1}}[x_1, \ldots, x_l]$. This is a contradiction to the preconditions. Consequently, $\mathcal{F}$ is linearly independent over the integers.
A direct application of Theorem 5.1.4 thus gives that $\mathcal{F}$ is determinant preserving. This concludes the proof. ∎

Remark 5.1.6
Note that a determinant preserving set only contains polynomials $f \in \mathbb{Z}[x_1, \ldots, x_l]$ with relatively prime coefficients. Any polynomial with $a > 1$ as greatest common divisor of the coefficients is the zero polynomial in $\mathbb{Z}_a[x_1, \ldots, x_l]$ and, therefore, linearly dependent

modulo a.

The above theorem gives us a condition on which sets of polynomials can be used. However, to use this condition we have to check linear independence for any $a \in \mathbb{N} \setminus \{1\}$ which, of course, cannot be done efficiently. Therefore, we would like to have only a selection of integers which we have to check. However, there is no such selection in general as the following example shows.

Example 5.1.7
Let

$$\mathbf{B} := \begin{pmatrix} -2 & -3 \\ 3 & 2 \\ 4 & 1 \end{pmatrix}.$$

By a unimodular transformation $\mathbf{U}$ *the matrix* $\mathbf{B}$ *can be transformed into*

$$\mathbf{UB} := \begin{pmatrix} 1 & 0 \\ 0 & 5 \\ 0 & 0 \end{pmatrix}.$$

Thus, the elementary divisors of $\mathbf{B}$ *are* 1 *and* 5*. Regarding the column vectors, we see that*

$$\begin{pmatrix} -3 \\ 2 \\ 1 \end{pmatrix} = 4 \begin{pmatrix} -2 \\ 3 \\ 4 \end{pmatrix} \pmod{5}.$$

That is, we do have a modular relation modulo 5 *between the two column vectors. The modulus* 5*, however, does not have any obvious relation to the matrix and its entries.*

Nevertheless, we do not have to check for all moduli $1 \neq a \in \mathbb{N}$. An upper bound on the number of potential moduli is given by $\left(\sqrt{|\mathcal{F}|} \cdot |c_{\max}|\right)^{|\mathcal{F}|}$, where $\mathcal{F} \subset \mathbb{Z}[x_1, \ldots, x_l]$ is the shift polynomial set we consider, and $c_{\max}$ is the largest coefficient which occurs in $\mathcal{F}$.
This can be seen as follows. Let again $\mathbf{F}$ be the matrix induced by $\mathcal{F}$. Combining the proofs of Theorem 5.1.4 and Theorem 2.1.3, we know that any potential modulus is an elementary divisor of $\mathbf{F}$. Therefore, to calculate an upper bound on the size of the potential moduli, we determine an upper bound on the size of the largest elementary divisor. Let $\mathbf{U}$ denote the unimodular transformation such that $\mathbf{UF}$ is the matrix with the elementary divisors on its diagonal and all other entries being equal to zero. Let $\tilde{\mathbf{UF}}$ denote the matrix $\mathbf{UF}$ without its zero rows. Then $\tilde{\mathbf{UF}}$ is an $|\mathcal{F}| \times |\mathcal{F}|$ submatrix of $\mathbf{UF}$. Moreover, $\det(\tilde{\mathbf{UF}})$ is the product of the elementary divisors. Let $\tilde{\mathbf{F}}$ denote an $|\mathcal{F}| \times |\mathcal{F}|$ submatrix of $\mathbf{F}$ with linearly independent rows. Then $\det(\tilde{\mathbf{UF}}) \leq \det(\tilde{\mathbf{F}})$. Furthermore,

$$\det(\tilde{\mathbf{F}}) \leq \prod_{i=1}^{|\mathcal{F}|} \left|\left|(\tilde{\mathbf{F}})_{i,\cdot}\right|\right| \leq \left(\sqrt{|\mathcal{F}|} \cdot |c_{\max}|\right)^{|\mathcal{F}|}.$$

Thus, we obtain an upper bound on the largest elementary divisor and, consequently, on the largest potential modulus by $\left(\sqrt{|\mathcal{F}|}\cdot|c_{\max}|\right)^{|\mathcal{F}|}$.

Unfortunately, all these moduli still cannot be tested efficiently. Whereas we require the algorithms to have running time polynomial in the bitsize of $c_{\max}$, the upper bound on the number of moduli we have to check is only polynomial in $c_{\max}$. That is, it is exponential in $\log(c_{\max})$.

In contrast to the matrix $\mathbf{B}$ given in the above example, the matrices we construct using Coppersmith's method have some special properties. While constructing the shift polynomial set, the polynomials are built in a way that every new polynomial introduces at least one new monomial. The part of the basis matrix corresponding to the shift polynomials is, thus, a matrix comprised of an upper triangular matrix and additional rows. Luckily, this property enables us to further restrict the number of moduli we have to check.

Lemma 5.1.8
Let $\mathcal{F} := \{f_1, \ldots, f_{|\mathcal{F}|}\} \subset \mathbb{Z}[x_1, \ldots, x_l]$ be an ordered set of polynomials constructed such that $Mon(f_j) \setminus \left(\cup_{i=1}^{j-1} Mon(f_i)\right) \neq \emptyset$ which is not determinant preserving. Let $\mathbf{F}$ be the matrix induced by this set as before. Then there are $1 \neq a \in \mathcal{C}$, $\bar{f} \in \mathcal{F}$ and $c_f \in \mathbb{Z}$ such that $\bar{f} \equiv \sum_{f\in\mathcal{F}\setminus\{\bar{f}\}} c_f f \pmod{a}$. Here, $\mathcal{C}$ denotes the set of all coefficients of monomials in $Mon(\mathcal{F})$ and divisors of them.

Proof: From the precondition using Theorem 5.1.5 it follows that there are $\bar{f} \in \mathcal{F}$, $c_f \in \mathbb{Z}$ for all $f \in \mathcal{F} \setminus \{\bar{f}\}$, and $1 \neq a \in \mathbb{N}$ such that $\sum_{f\in\mathcal{F}\setminus\{\bar{f}\}} c_f f \equiv \bar{f} \pmod{a}$. Let $w := |\mathrm{Mon}(\mathcal{F})|$ and let $m_1, \ldots, m_{|\mathrm{Mon}(\mathcal{F})|}$ be an enumeration of the monomials of $\mathrm{Mon}(\mathcal{F})$ corresponding to the construction of $\mathbf{F}$. Let h be the largest index such that the monomial m_h occurs in the equation with non-zero coefficient. Due to the upper triangular structure, this monomial only occurs in one of the polynomials in the equation. (Otherwise, assume it occurs in two polynomials f_i and f_j with $j > i$. Then f_j must introduce another monomial m_2 corresponding to a row index greater than h, which contradicts the choice of h.) Let $c(m_h)$ be the coefficient of m_h. Then for the equation to hold we require $a|c(m_h)$. Thus, $a \in \mathcal{C}$ which implies the claim. ■

This lemma will be quite helpful in the following analyses as the polynomials we deal with have coefficients which are mostly products of only two primes. This can be seen in Section 6.1.
But, of course, we then have to construct a shift polynomial set in a way that it has the required structure.

Before we turn to the analysis of various systems of equations, we will extend the definition of being determinant preserving. So far we have dealt with polynomial sets $\mathcal{F} \subset \mathbb{Z}[x_1, \ldots, x_l]$ to which the definition can be applied. In practice, we also have polynomial equations $\tilde{f}(x_1, \ldots, x_l) \equiv 0 \pmod{N}$. That is, we would like to determine solutions modulo an integer N. That is, the polynomial $\tilde{f}(x_1, \ldots, x_l)$ is a polynomial in $\mathbb{Z}_N[x_1, \ldots, x_l]$.

To include various modular equations with equal or different moduli into our set, let $\tilde{\mathcal{F}}$ denote the set of all shift polynomials and their corresponding moduli. Elements of $\tilde{\mathcal{F}}$ are denoted by tuples $(\tilde{f}(x_1, \ldots, x_l), N)$. Each polynomial $\tilde{f}(x_1, \ldots, x_l) \in \mathbb{Z}_N[x_1, \ldots, x_l]$ can be written as a polynomial in a polynomial ring over the integers with one more unknown. The corresponding integer polynomial is $f(x_1, \ldots, x_l, t) := \tilde{f}(x_1, \ldots, x_l) - tN \in \mathbb{Z}[x_1, \ldots, x_l, t]$. We call the polynomial f the integer polynomial induced by $\tilde{f}$. Accordingly, we define the set $\mathcal{F}$ to consist of all integer polynomials f induced by polynomials $\tilde{f} \in \tilde{\mathcal{F}}$. The set $\mathcal{F}$ is then called the polynomial set induced by $\tilde{\mathcal{F}}$. Note that for each polynomial in $\tilde{\mathcal{F}}$ one new variable is adjoint to the polynomial ring. Thus, $\mathcal{F} \subset \mathbb{Z}[x_1, \ldots, x_l, t_1, \ldots, t_{|\tilde{\mathcal{F}}|}]$. With this notation we can now extend the definition of determinant preserving sets to modular polynomials.

Definition 5.1.9
Let the set of polynomials $\tilde{\mathcal{F}}$ consist of tuples of polynomials and their corresponding moduli. Then the set $\tilde{\mathcal{F}}$ is called a ***determinant preserving set*** *if and only if the set $\mathcal{F} \subset \mathbb{Z}[x_1, \ldots, x_l, t_1, \ldots, t_{|\tilde{\mathcal{F}}|}]$ induced by $\tilde{\mathcal{F}}$ is determinant preserving.*

With this definition we can derive similar conditions like the one of Lemma 5.1.8 for systems of modular polynomial equations. As these conditions differ depending on the relation of the moduli to each other, they will be presented in the corresponding sections.

5.2 Systems of Equations with a Common Modulus

In this section we analyze systems of modular multivariate polynomial equations with a common modulus N, namely, we refer to a special case of the SMMPE problem. In analogy to the univariate case, we define the problem of solving *systems of modular multivariate polynomial equations with a common modulus* (SMMPE1-problem).

Definition 5.2.1 (SMMPE1-problem)
Let $k \in \mathbb{N}$, $\delta_1, \ldots, \delta_k \in \mathbb{N}$, and $N \in \mathbb{N}$. Assume $f_1(x_1, \ldots, x_l), \ldots, f_k(x_1, \ldots, x_l)$ to be polynomials of degree $\delta_1, \ldots, \delta_k$ in $\mathbb{Z}_N[x_1, \ldots, x_l]$, respectively. Let

$$\begin{array}{rcl} f_1(x_1, \ldots, x_l) & \equiv & 0 \pmod{N} \\ f_2(x_1, \ldots, x_l) & \equiv & 0 \pmod{N} \\ & \vdots & \\ f_k(x_1, \ldots, x_l) & \equiv & 0 \pmod{N} \end{array} \tag{5.1}$$

be a system of multivariate polynomial equations.

Let $X_i < N$, $X_i \in \mathbb{R}$, $i = 1, \ldots, l$. Find all common roots $(\bar{x}_1, \ldots, \bar{x}_l)$ of (5.1) with size $|\bar{x}_i| \leq X_i$.

The analysis of modular systems is simpler than in the integer case. Let $\tilde{\mathcal{F}}$ be a set of shift polynomials and their respective moduli used to apply Coppersmith's method 2.3.9.

Then all $f \in \tilde{\mathcal{F}}$ are polynomials of the form $mf_i(x_1, \ldots, x_l)^\lambda$ with $m \in \mathbb{Z}_N[x_1, \ldots, x_l]$ being some monomial and $\lambda \in \mathbb{N}$. Thus, the solution we are searching for is a solution of $f \equiv 0 \pmod{N^\lambda}$ for all $f \in \tilde{\mathcal{F}}$.

Let $\mathcal{F}$ be the set of integer polynomials induced by $\tilde{\mathcal{F}}$. From these polynomials a basis matrix is constructed. The last columns of this basis matrix correspond to the shift polynomials. Let us again denote this matrix by $\mathbf{F}$ as it is done in Section 2.3. It consists of an upper part $\mathbf{F_c}$ corresponding to the coefficients of the polynomials and a lower part $\mathbf{F_m}$ corresponding to the moduli. The matrix $\mathbf{F_m}$, thus, forms a diagonal matrix with entries which are powers of N.

In the analysis in Section 5.1 we have seen additional conditions which allow for a simpler verification if $\mathcal{F}$ is determinant preserving. Namely, if $\mathbf{F}$ consists of some upper triangular matrix and some arbitrary matrix, the check becomes easier. In this example the upper triangular part of $\mathbf{F}$ is given by $\mathbf{F_m}$. The matrix $\mathbf{F_m}$ is not only upper triangular, but even a diagonal matrix with the moduli as values on the diagonal. The moduli are always powers N^λ for a value of $\lambda \in \mathbb{N}$. Therefore, the only way to have linear dependence of vectors of $\mathbf{F}$ is to have dependence modulo N or a divisor of N. Note that linear dependence modulo N^λ with $\lambda > 1$ implies linear dependence modulo N. Hence, it is sufficient to test for modular dependence with regard to N and its divisors.

Lemma 5.2.2

Let $\tilde{\mathcal{F}} \subset \mathbb{Z}[x_1, \ldots, x_l] \times \mathbb{N}$ be an ordered set of polynomials and corresponding moduli which is not determinant preserving. All moduli occurring in $\tilde{\mathcal{F}}$ are powers of N. Let $\mathcal{F}$ be the integer polynomial set induced by $\tilde{\mathcal{F}}$, and $\mathbf{F}$ be the matrix induced by $\mathcal{F}$ as before. Then there are $\bar{f} \in \mathcal{F}$ and $c_f \in \mathbb{Z}$ such that $\bar{f} \equiv \sum_{f \in \mathcal{F} \setminus \{\bar{f}\}} c_f f \pmod{a}$, where $a|N$ and $a > 1$.

Proof: We regard the polynomials $f \in \mathcal{F} \subset \mathbb{Z}[x_1, \ldots, x_l, t_1, \ldots, t_{|\mathcal{F}|}]$ induced by polynomials $\tilde{f} \in \tilde{\mathcal{F}}$. From the precondition using Theorem 5.1.5 it follows that there are $\bar{f} \in \mathcal{F}$, $c_f \in \mathbb{Z}$ for all $f \in \mathcal{F} \setminus \{\bar{f}\}$, and $1 \neq a \in \mathbb{N}$ such that $\sum_{f \in \mathcal{F} \setminus \{\bar{f}\}} c_f f \equiv \bar{f} \pmod{a}$. By construction there is a monomial t_j which occurs only in $\bar{f}$. Its coefficient is N^λ for some $\lambda \in \mathbb{N}$ as it corresponds to the modulus. Then for the equation to hold we require $a|N^\lambda$ which implies the claim. ■

In many practical examples we take moduli N which are products of two unknown primes p and q. Consequently, we can only test for linear dependence modulo N, but not modulo its divisors. Nevertheless, this does not pose any problems. Linear dependence modulo p or q implies that we get p or q as elementary divisor of the matrix. Thus, a way to break the system would be to take some set of polynomials which is not determinant preserving, to build up the corresponding matrix, and to calculate its elementary divisors. This can be done quite efficiently (compare [Lüb02]). Then, given p or q, the system from which we derived the equations is broken anyway.

In the following examples we, therefore, assume that we do not get any factors of the moduli as elementary divisors. This assumption was valid in all our examples.

5.2.1 RSA with Related Messages and Implicit Relations

We recall the problem of RSA with related messages and implicit relations presented in Section 4.1. A set of k secret messages $m_1, \dots, m_k$ is encrypted with an RSA public key (N, e). Each encryption corresponds to an equation $f_i(x_i) := x_i^e - c_i \equiv 0 \pmod{N}$ with a solution m_i. Furthermore, the messages are related by some implicit polynomial relation $p(x_1, \dots, x_k)$ such that $p(m_1, \dots, m_k) \equiv 0 \pmod{N}$. We have already seen how we can determine all solutions of the system (Section 4.1) by computing a Groebner basis. Now we will show a method to determine certain roots if Groebner basis computations do not help. For ease of analysis we start with the case of $k = 2$ and a simple lattice basis. The cases $k > 2$ or more complex lattice bases can be treated analogously. Moreover, we assume to get a relation p to be valid over the integers. That is, we regard the following problem:

$$\begin{array}{rcll} f_1(x_1) & \equiv & 0 & \pmod{N} \\ f_2(x_2) & \equiv & 0 & \pmod{N} \\ p(x_1, x_2) & = & 0\,. & \end{array} \tag{5.2}$$

Let $X_i < N$, $X_i \in \mathbb{R}$, $i = 1, 2$. Find all common roots (m_1, m_2) such that $|m_i| \leq X_i$.

Thus, we look at the SMMPE1-problem with a simple system of multivariate equations. For $i \in \{1, 2\}$ let us first regard $f_i(x_i) \equiv 0 \pmod{N}$ separately. We deal with a modular univariate equation. We can determine its solutions $|m_i| \leq X_i$ if $X_i < N^{\frac{1}{e}}$ in polynomial time according to Theorem 2.3.9. The initial lattice basis used is

$$\mathbf{B}^i = \begin{pmatrix} \mathbf{D}^i & \mathbf{F_c}^i \\ \mathbf{0} & \mathbf{F_m}^i \end{pmatrix} = \begin{pmatrix} 1 & 0 & -c_i \\ 0 & \frac{1}{X_i^e} & 1 \\ 0 & 0 & N \end{pmatrix}. \tag{5.3}$$

As the polynomials $f_i(x_i)$ only consist of the monomials x_i^e and 1, we just put f_i into the shift polynomial set. Then the bound can directly be computed using the simplified condition $\det(\mathbf{B}^i) > 1$ given in equation (2.14).

In a first step to combine the analyses, we combine the basis matrices. That is, at the moment we ignore the additional information we get by the additional polynomial p, but only consider the original problem. We set

$$\mathbf{B} = \begin{pmatrix} 1 & 0 & 0 & -c_1 & -c_2 \\ 0 & \frac{1}{X_1^e} & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{X_2^e} & 0 & 1 \\ 0 & 0 & 0 & N & 0 \\ 0 & 0 & 0 & 0 & N \end{pmatrix} \tag{5.4}$$

as initial basis matrix for a lattice L to determine solutions of both equations $f_i(x_i) \equiv 0 \pmod{N}$ in one analysis. We apply the simplified condition to this and obtain

$$\det(\mathbf{B}) > 1 \;\Leftrightarrow\; \det(\mathbf{B}^1)\det(\mathbf{B}^2) > 1\,.$$

This condition is, of course, fulfilled if the basic conditions

$$\det(\mathbf{B}^1) > 1 \text{ and } \det(\mathbf{B}^2) > 1 \tag{5.5}$$

hold. Thus, a sufficient condition that we can determine the solutions (m_1, m_2) using the lattice basis $\mathbf{B}$ given in (5.4) in Coppersmith's method is that we can determine it separately using the lattice bases $\mathbf{B}^1$ and $\mathbf{B}^2$ described in (5.3), i. e. condition (5.5) holds. As the equations $f_1(x_1) \equiv 0 \pmod{N}$ and $f_2(x_2) \equiv 0 \pmod{N}$ do not share any variables, condition (5.5) ought to be necessary as well. However, just regarding the determinant, this is obviously not the case. If e. g. $|\det(\mathbf{B}^1)| \gg 1$, then $|\det(\mathbf{B}^2)|$ could be smaller than one without violating condition (5.5) anyway. To see why Coppersmith's method fails under these preconditions, we look at it in more detail. By failure in this case we mean that we do not get any further equations containing the variable x_2. Thus, we are only able to determine m_1, but not m_2. Note, however, that we can calculate m_2 by substituting x_1 by m_1 in $p(x_1, x_2)$ and then determining the roots of the univariate polynomial $p(m_1, x_2) \in \mathbb{Z}[x_2]$ over the integers. This works in this case as only one message is missing. In case of $k > 2$, however, we could fail to determine more than one message. In order to develop a general strategy which can be applied to those cases as well, we continue with our example without taking p into account.
Let us now assume $|\det(\mathbf{B})| > 1$ but $|\det(\mathbf{B}^2)| \leq 1$. The latter implies $X_2^e \geq N$. We start with the basis $\mathbf{B}$ as basis of our lattice. In order to get a basis of the sublattice, the basis matrix $\mathbf{B}$ is transformed into

$$\mathbf{UB} = \begin{pmatrix} 1 & \frac{c_1}{X_1^e} & \frac{c_2}{X_2^e} & 0 & 0 \\ 0 & -\frac{N}{X_1^e} & 0 & 0 & 0 \\ 0 & 0 & -\frac{N}{X_2^e} & 0 & 0 \\ 0 & \frac{1}{X_1^e} & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{X_2^e} & 0 & 1 \end{pmatrix} \tag{5.6}$$

by a unimodular transformation $\mathbf{U}$. Let L_S denote the sublattice of vectors with zeros in their last two components. Then a basis $\mathbf{B_S}$ of L_S is given by the first three rows of $\mathbf{B}$. When considering vectors in L_S, we regard them as vectors with only three components as the other components are zero anyway. Recall that L_S contains the vector $\mathbf{t_S} := (1, \frac{m_1^e}{X_1^e}, \frac{m_2^e}{X_2^e})$ with norm $||\mathbf{t_S}|| \leq \sqrt{3} \in \mathcal{O}(1)$.
Subsequently, we perform LLL-reduction on $\mathbf{B_S}$ and orthogonalize it. We denote the resulting matrix by $\mathbf{B_R}^*$. Let $\mathbf{b_1}^*, \mathbf{b_2}^*, \mathbf{b_3}^*$ be the rows of $\mathbf{B_R}^*$. Due to the lower bound on the absolute value of the determinant, the vector $\mathbf{t_S}$ is at least orthogonal to $\mathbf{b_3}^*$. That is, by construction we have $||\mathbf{t_S}|| < ||\mathbf{b_i}^*||$ for at least $i = 3$. It might also hold for further basis vectors. Each such vector $\mathbf{b_i}^*$ is orthogonal to $\mathbf{t_S}$. Using this condition, we get a new non-modular equation $(\mathbf{b_i}^*)_1 + (\mathbf{b_i}^*)_2 \cdot \frac{m_1^e}{X_1^e} + (\mathbf{b_i}^*)_3 \cdot \frac{m_2^e}{X_2^e} = 0$. The solutions m_1 and m_2 are, thus, integer solutions of the equation $(\mathbf{b_i}^*)_1 + (\mathbf{b_i}^*)_2 \cdot \frac{x_1^e}{X_1^e} + (\mathbf{b_i}^*)_3 \cdot \frac{x_2^e}{X_2^e} = 0$.
We will show in the following paragraph that $(\mathbf{b_i}^*)_3 = 0$ for all $\mathbf{b_i}^*$ orthogonal to $\mathbf{t_S}$. This

implies that we only get additional information in x_1 and still cannot determine a solution of $f_2(x_2)$. Looking at the basis matrix, we can easily see that the vector $\mathbf{v} := (0, 0, -\frac{N}{X_2^e})$ also is a vector in L_S. From the condition $X_2^e \geq N$ we derive $|-\frac{N}{X_2^e}| \leq 1$ and, thus, $||\mathbf{v}|| \leq 1$. We even have $||\mathbf{v}|| \leq ||\mathbf{t_s}||$. Therefore, $\mathbf{v}$ is a vector in the same hyperplane as $\mathbf{t_S}$ is. Consequently, any vector $\mathbf{b_i}^*$ orthogonal to that hyperplane is also orthogonal to $\mathbf{v}$. Hence, we get the condition $(\mathbf{b_i}^*)_1 \cdot 0 + (\mathbf{b_i}^*)_2 \cdot 0 - (\mathbf{b_i}^*)_3 \cdot \frac{N}{X_2^e} = 0$. This implies $(\mathbf{b_i}^*)_3 = 0$. We generalize this observation to an arbitrary number of k equations and show that combining shift monomial sets does not improve the bound.

Theorem 5.2.3
Let $k \in \mathbb{N}$ and $f_i(x_i) \equiv 0 \pmod N$, $i = 1, \ldots, k$, be polynomials of degree δ_i. Let $\bar{x}_i$ denote a root of f_i, i. e. $f_i(\bar{x}_i) \equiv 0 \pmod N$. Let $X_i \in \mathbb{R}$ be a bound on the solution $\bar{x}_i$, namely $|\bar{x}_i| \leq X_i$. Then the following holds: If all solutions $(\bar{x}_1, \ldots, \bar{x}_k) \in \mathbb{Z}_N^k$ can be determined applying Coppersmith's method to one lattice built with regard to a combined shift polynomial set $\mathcal{F} = \cup_{i=1}^k \mathcal{F}_i$, where $\mathcal{F}_i \subset \mathbb{Z}_N[x_i]$ denotes a shift polynomial set with respect to f_i, then we can determine the solutions using separate lattices with regard to $\mathcal{F}_i$ as well. Furthermore, $X_i < N^{\frac{1}{\delta_i}}$ for all i.

Proof: First recall that Theorem 2.3.9 states that for any i the equation $f_i(\bar{x}_i) \equiv 0 \pmod N$ can be solved if $|\bar{x}_i| \leq X_i < N^{\frac{1}{\delta_i}}$. The shift polynomial set used for the analysis is $\mathcal{F}_i \subseteq \mathbb{Z}_N[x_i]$. Let $\mathcal{F} := \cup_{i=1}^k \mathcal{F}_i$. The set $\mathrm{Mon}(\mathcal{F})$ denotes the set of all monomials occurring in $\mathcal{F}$. We denote by $w_i := |\mathrm{Mon}(\mathcal{F}_i)|$ and by $w := |\mathrm{Mon}(\mathcal{F})|$ the number of monomials occurring in $\mathcal{F}_i$ and $\mathcal{F}$, respectively. We will show that if all solutions $(\bar{x}_1, \ldots, \bar{x}_k) \in \mathbb{Z}_N^k$ can be determined applying Coppersmith's method with shift polynomial set $\mathcal{F}$, then they can be determined applying Coppersmith's method to each polynomial f_i separately using $\mathcal{F}_i$. This results in the claim.

Without loss of generality we assume all $f \in \mathcal{F}$ to be monic. If they are not, we can either invert the leading coefficient modulo N or compute a divisor of N. By Theorem 2.3.9 it is $f_i \in \mathcal{F}_i$. Let

$$\mathbf{B}^i = \begin{pmatrix} \mathbf{D}^i & \mathbf{F_c}^i \\ \mathbf{0} & \mathbf{F_m}^i \end{pmatrix} = \begin{pmatrix} 1 \;\; 0 \;\; \ldots \;\; 0 & f_{i0} \;\; * \;\; \ldots \;\; * \\ \begin{matrix} 0 & \tilde{\mathbf{D}}^i \end{matrix} & \tilde{\mathbf{F_c}}^i \\ \mathbf{0} & \mathbf{F_m}^i \end{pmatrix} \tag{5.7}$$

denote a basis matrix of the lattice L_i constructed with regard to $\mathcal{F}_i$ to obtain this bound. In the second representation of the initial basis matrix $\mathbf{B}^i$, the row corresponding to the monomial 1 is explicitly named so that the matrix $\tilde{\mathbf{D}}^i$ equals $\mathbf{D}^i$ without its first row and column. The matrix $\tilde{\mathbf{F_c}}^i$ equals $\mathbf{F_c}^i$ without its first row. The value $f_{i0} \in \mathbb{Z}_N$ denotes the constant term of f_i. The values $*$ correspond to the constant terms of the other shift polynomials. They might also be equal to zero. A basis of the lattice L constructed with regard to $\mathcal{F}$ is given by

$$\mathbf{B} = \left(\begin{array}{c|c|c|c}
1 \quad 0 \quad \dots \quad 0 & f_{10} \quad * \quad \dots \quad * & \dots & f_{k0} \quad * \quad \dots \quad * \\ \hline
\begin{matrix} \tilde{\mathbf{D}}^1 & & \\ & \ddots & \\ & & \tilde{\mathbf{D}}^k \end{matrix} & \begin{matrix} \tilde{\mathbf{F}_\mathbf{c}}^1 \\ \\ \\ \end{matrix} & \ddots & \begin{matrix} \\ \\ \tilde{\mathbf{F}_\mathbf{c}}^k \end{matrix} \\ \hline
 & \mathbf{F_m}^1 & \ddots & \\ \hline
 & & & \mathbf{F_m}^k
\end{array}\right). \tag{5.8}$$

Non-included values are equal to zero.
We use this notation to prove the claim by contraposition. We show "If $X_i > N^{\frac{1}{\delta_i}}$ for some $i \in \{1, \dots, k\}$ (that is, if we cannot determine $\bar{x}_i$ via Coppersmith's algorithm using the lattice L_i), then we cannot determine $\bar{x}_i$ using Coppersmith's algorithm with basis matrix $\mathbf{B}$ either".
Without loss of generality we assume $X_1 > N^{\frac{1}{\delta_1}}$. (In any other case we simply permute the polynomials in the shift polynomial set.)
The sets $\mathcal{F}_i$ are determinant preserving by construction. As each set contains a different variable x_i, their union $\mathcal{F}$ is determinant preserving as well. For a proof let us assume the contrary. Then there is a subset $\mathcal{S} \subseteq \mathcal{F}$ with $\mathcal{S} \cap \mathcal{F}_i \neq \emptyset$ for at least two indices $i \in \{1, \dots, k\}$ such that $\sum_{f \in \mathcal{S}} c_f f \equiv 0 \pmod{N}$ and $c_f \neq 0$. For $i \in \{1, \dots, k\}$ with $\mathcal{F}_i \cap \mathcal{S} \neq \emptyset$ let $\bar{f}_i$ denote the polynomial of maximal degree in x_i. Let $x_i^{\Delta_i}$ denote the monomial of maximum degree. The polynomial $\bar{f}_i$ is unique as any polynomial in $\mathcal{F}_i$ (if the polynomials are ordered appropriately) introduces a new monomial. Therefore, no other polynomial in $\mathcal{F}_i \cap \mathcal{S}$ contains the monomial $x_i^{\Delta_i}$. Furthermore, polynomials in $\mathcal{F}_j$ with $j \neq i$ do not contain monomials in x_i. Consequently, $\bar{f}_i \in \mathcal{S}$ is the only polynomial in which the monomial $x_i^{\Delta_i}$ occurs. For the linear dependence relation to hold, we need to have $c_{\bar{f}_i} \equiv 0 \pmod{N}$. This contradicts the assumption that $\bar{f}_i \in \mathcal{S}$. Thus, $\mathcal{F}$ is determinant preserving. This implies that there exists a unimodular transformation of $\mathbf{B}$ to

$$\begin{pmatrix} \mathbf{B_S} & \mathbf{0} \\ * & \mathbf{I}_{|\mathcal{F}|} \end{pmatrix}. \tag{5.9}$$

The matrix $*$ is a matrix with arbitrary values which are not important for our analysis. We denote by L_S the lattice spanned by the basis vectors of $\mathbf{B_S}$. Let us look at $\mathbf{B_S}$ in more detail. Recall that large vectors of the LLL-reduced and orthogonalized version of $\mathbf{B_S}$ are orthogonal to some target vector $\mathbf{t}$. This gives rise to additional equations in the unknowns. Our goal is to show that these vectors have a special structure so that the induced equations do not contain monomials in x_1. Then we cannot get further information on $\bar{x}_1$ and are, therefore, unable to determine it this way. Due to the structure of $\mathbf{B}$ we can look at the matrix

$$\tilde{\mathbf{B}}^1 = \begin{pmatrix} \tilde{\mathbf{D}}^1 & \tilde{\mathbf{F}_\mathbf{c}}^1 \\ \mathbf{0} & \mathbf{F_m}^1 \end{pmatrix} \tag{5.10}$$

separately. What we do here is taking the second until (w_1)-th and the $(w+1)$-th until $(w+|\mathcal{F}_1|)$-th basis vector. These basis vectors only have non-zero entries in the second until (w_1)-th and the $(w+1)$-th until $(w+|\mathcal{F}_1|)$-th columns. Therefore, it suffices to regard those. Operations on the rows of $\tilde{\mathbf{B}}^1$ do not influence any other columns. As the polynomials in $\mathcal{F}_1$ are monic and each new polynomial introduces a new monomial with coefficient one, $\tilde{\mathbf{F_c}}^1$ contains a diagonal with values one. The values below this diagonal are equal to zero. The rows corresponding to the ones can be permuted with the rows of $\mathbf{F_m}^1$. Then, we have transformed $\tilde{\mathbf{B}}^1$ to

$$\begin{pmatrix} \tilde{\mathbf{D}}_u^1 & \tilde{\mathbf{F_c}}_u^1 \\ \mathbf{0} & \mathbf{F_m}^1 \\ \tilde{\mathbf{D}}_l^1 & \tilde{\mathbf{F_c}}_l^1 \end{pmatrix}. \tag{5.11}$$

Here, $\tilde{\mathbf{D}}_l^1$ denotes the lower $|\mathcal{F}_1|$ rows of $\tilde{\mathbf{D}}^1$ and $\tilde{\mathbf{D}}_u^1$ its upper part. Analogously, $\tilde{\mathbf{F_c}}_l^1$ denotes the lower $|\mathcal{F}_1|$ rows of $\tilde{\mathbf{F_c}}^1$ and $\tilde{\mathbf{F_c}}_u^1$ its upper part. The last $|\mathcal{F}_1|$ columns form a rectangular matrix. Its last $|\mathcal{F}_1|$ rows form an upper triangular matrix with value one on the diagonal. To conclude, we use these last rows to eliminate all further values. Thus, by further transformations, we get

$$\begin{pmatrix} \mathbf{B_S}^1 & \mathbf{0} \\ * & \mathbf{I}_{|\mathcal{F}_1|} \end{pmatrix}. \tag{5.12}$$

Note that each row vector is $(w_1 - 1 + |\mathcal{F}_1|)$-dimensional as the first column, which corresponds to constant terms, is not included. We are interested in the values in $\mathbf{B_S}^1$ because they correspond to row vectors in $\mathbf{B_S}$ as it is

$$\mathbf{B_S} = \begin{pmatrix} * & & * & \\ & \mathbf{B_S}^1 & & \\ \mathbf{0} & & \ddots & \\ & & & \mathbf{B_S}^k \end{pmatrix}. \tag{5.13}$$

A row of $\begin{pmatrix} \mathbf{B_S}^1 & \mathbf{0} \end{pmatrix}$ is constructed in one of two ways. Either it originates from a row of $\begin{pmatrix} \tilde{\mathbf{D}}_u^1 & \tilde{\mathbf{F_c}}_u^1 \end{pmatrix}$ or it originates from a row of $\begin{pmatrix} \mathbf{0} & \mathbf{F_m}^1 \end{pmatrix}$. For $d = 1, \ldots, \delta_1 - 1$ let $\tilde{\mathbf{u}}_\mathbf{d} := (0, \ldots, 0, \frac{1}{X_1^d}, 0, \ldots, 0, c_1, \ldots, c_{|\mathcal{F}_1|}) = \frac{1}{X_1^d}\mathbf{e}^d + \sum_{i=1}^{|\mathcal{F}_1|} c_i \mathbf{e}^{w_1-1+i}$ denote a row in the first case. In this case all non-zero values c_i have to be eliminated. Let $\mathbf{u_d} := (0, \ldots, 0, N, 0, \ldots, 0) = N\mathbf{e}^{w_1-1+d}$, $d = 1, \ldots, |\mathcal{F}_1|$, denote a row vector in the second case. Here only the value N has to be eliminated.
The question is what happens in the elimination step. The rows used for elimination are of type $\tilde{\mathbf{v}}_{\mathbf{w_1-1+d}} := (0, \ldots, 0, \frac{1}{X_1^{\delta_1-1+d}}, 0, \ldots, 0, 1, a_{w_1+d}^{(d)}, \ldots, a_{w_1+|\mathcal{F}_1|}^{(d)}) = \frac{1}{X_1^{\delta_1-1+d}}\mathbf{e}^{\delta_1-1+d} + \mathbf{e}^{w_1-1+d} + \sum_{j=d}^{|\mathcal{F}_1|-1} a_{w_1+j}^{(d)} \mathbf{e}^{w_1+j}$ for $d = 1, \ldots, |\mathcal{F}_1|$ with $a_{w_1+j}^{(d)} \in \mathbb{Z}$. In a first step, we transform these vectors by eliminating the $a_{w_1+j}^{(d)}$. We set $\mathbf{v}_{\mathbf{w_1-1+|\mathcal{F}_1|}} := \tilde{\mathbf{v}}_{\mathbf{w_1-1+|\mathcal{F}_1|}}$ and iteratively define $\mathbf{v}_{\mathbf{w_1-1+d}} := \tilde{\mathbf{v}}_{\mathbf{w_1-1+d}} - a_{w_1+d}^{(d)} \mathbf{v}_{\mathbf{w_1-1+(d+1)}}$ for $d = |\mathcal{F}_1| - 1, \ldots, 1$. Thus,

$\mathbf{v_{w_1-1+d}} := \sum_{j=d}^{|\mathcal{F}_1|} \frac{(-1)^{j-d}\prod_{s=d+1}^{j} a_{w_1+s-1}^{(s-1)}}{X_1^{\delta_1-1+j}} \mathbf{e}^{\delta_1-1+j} + \mathbf{e}^{w_1-1+d}$.

We start by modifying the vectors $\mathbf{u_d}$. To eliminate N, one has to compute $\tilde{\mathbf{b}}_{\delta_1-1+\mathbf{d}} := \mathbf{u_d} - N\mathbf{v_{w_1-1+d}}$, $d = 1, \dots, |\mathcal{F}_1|$, and gets a vector with zeros as last $|\mathcal{F}_1|$ values because it is $w_1 - 1 \geq |\mathcal{F}_1| + \delta_1 - 1$. The first values, thus, give a row vector in $\mathbf{B_S}^1$, namely $\sum_{j=d}^{|\mathcal{F}_1|} \frac{(-1)^{j-d+1} N \prod_{s=d+1}^{j} a_{w_1+s-1}^{(s-1)}}{X_1^{\delta_1-1+j}} \mathbf{e}^{\delta_1-1+j}$. Note that the $(\delta_1 - 1 + d)$-th component of $\tilde{\mathbf{b}}_{\delta_1-1+\mathbf{d}}$ is $\frac{-N}{X_1^{\delta_1-1+d}}$. It has absolute value smaller than one because due to the preconditions we have $X_1^d \geq X_1^{\delta_1} > N$ if $d \geq \delta_1$. Successively substracting appropriate multiples of $\tilde{\mathbf{b}}_{\delta_1-1+\mathbf{j}}, j = d+1, \dots, |\mathcal{F}_1|$, from $\tilde{\mathbf{b}}_{\delta_1-1+\mathbf{d}}$, we get a vector $\mathbf{b}_{\delta_1-1+\mathbf{d}}$ such that alle entries are smaller than one. Consequently, an upper bound on the norm of $\mathbf{b}_{\delta_1-1+\mathbf{d}}$ is given by $\sqrt{|\mathcal{F}_1| + 1}$.
To eliminate the last entries of $\tilde{\mathbf{u}}_\mathbf{d}$, we subtract c_j-times $\mathbf{v_{w_1-1+j}}$ from $\tilde{\mathbf{u}}_\mathbf{d}$. This results in $\tilde{\mathbf{b}}_\mathbf{d} := \frac{1}{X_1^d}\mathbf{e}^d + \sum_{j=1}^{|\mathcal{F}_1|} -c_j \left(\sum_{j=d}^{|\mathcal{F}_1|} \frac{(-1)^{j-d}\prod_{s=d+1}^{j} a_{w_1+s-1}^{(s-1)}}{X_1^{\delta_1-1+j}} \right) \mathbf{e}^{\delta_1-1+j}$ with $d = 1, \dots, \delta_1 - 1$. As $w_1 - 1 \geq |\mathcal{F}_1| + \delta_1 - 1$, the last $|\mathcal{F}_1|$ values in this vector are zero. Its first entries, thus, give a row vector in $\mathbf{B_S}^1$. We use the vectors $\mathbf{b}_{\delta_1-1+\mathbf{d}}$ for further reducing the other entries and denote the result by $\mathbf{b_d}$. As before all non-zero values in the reduced vector $\mathbf{b_d}$ are smaller than one. There are at most $|\mathcal{F}_1| + 1$ non-zero values in $\mathbf{b_d}$. Hence, $||\mathbf{b_d}|| \leq \sqrt{|\mathcal{F}_1| + 1}$.
Overall,

$$\begin{pmatrix} \mathbf{B_S}^1 & \mathbf{0} \end{pmatrix} = \begin{pmatrix} \mathbf{b_1} \\ \vdots \\ \mathbf{b}_{\delta_1-1+|\mathcal{F}_1|} \end{pmatrix}. \tag{5.14}$$

Recall that when applying Coppersmith's technique with regard to the lattice L using the basis $\mathbf{B}$, the target vector is defined as $\mathbf{t} := (1, \frac{x_1}{X_1}, \dots, \frac{x_1^{\delta_1+|\mathcal{F}_1|-1}}{X_1^{\delta_1+|\mathcal{F}_1|-1}}, \frac{x_2}{X_2}, \dots, \frac{x_k^{\delta_k+|\mathcal{F}_k|-1}}{X_k^{\delta_k+|\mathcal{F}_k|-1}})$. The target vector is of size $||\mathbf{t}|| \leq \sqrt{w}$. As each polynomial introduces at least one new monomial, we have $w = |\text{Mon}(\mathcal{F})| \geq |\mathcal{F}| \geq |\mathcal{F}_1| + 1$. Let $\mathbf{B}^*$ denote an LLL-reduced and orthogonalized basis of the sublattice L_S. If $||\mathbf{b_i^*}|| > ||\mathbf{t}||$ for an index i, then also $||\mathbf{b_i^*}|| > ||\mathbf{b_d}||$ for all $\mathbf{b_d}$. The vectors $\mathbf{b_d}$ are $\delta_1 - 1 + |\mathcal{F}_1|$ vectors in a vector subspace of dimension $\delta_1 - 1 + |\mathcal{F}_1|$. As the determinant of $\tilde{\mathbf{B}}^1$ is not equal to zero, the vectors are linearly independent and form a basis of this vector subspace. Another basis of this vector subspace is given by the set $\{\mathbf{e}^2, \dots, \mathbf{e}^{\delta_1+|\mathcal{F}_1|}\}$. Thus, if $\mathbf{b_i}^*$ is orthogonal to all $\mathbf{b_d}$, it is also orthogonal to $\mathbf{e}^i$ for $i = 2, \dots, \delta_1 + |\mathcal{F}_1|$. Consequently, $(\mathbf{b_i}^*)_d = 0$ with $d = 2, \dots, \delta_1 + |\mathcal{F}_1|$. The values $(\mathbf{b_i}^*)_d$ correspond to the second until w_1-th component of a vector in the original lattice L_S. Therefore, any equation we get by applying Coppersmith's method will not contain any monomial in x_1. This implies that we cannot determine $\bar{x}_1$ which concludes the proof. ∎

In the analysis of a system of independent equations, i. e. equations not sharing any variables, it is useless to construct a lattice like the lattice L to combine the analyses of the equations. This matches the results we have expected beforehand. Another option would be to shift the single polynomials by monomials in other variables. Advantage could then

be taken of shared monomials. The shared monomial of smallest degree when shifting f_1 and f_2 is the monomial $x_1^e x_2^e$. We obtain this monomial by the shift polynomials $x_1^e f_2$ and $x_2^e f_1$. However, it is $x_1^e f_2 - c_1 f_2 - x_2^e f_1 + c_2 f_1 \equiv f_1 f_2 - f_1 f_2 \equiv 0 \pmod{N}$. Hence, a shift polynomial set including the shifts $\{x_1^e f_2, f_2, x_2^e f_1, f_1\}$ is not determinant preserving. At least one of the polynomials has to be excluded from the shift polynomial set. Then the contribution of the shift polynomial set to the determinant is $X_1^{-2e} X_2^{-2e} N^3$. Without the "mixed" shifts the bound is $X_1^{-e} X_2^{-e} N^2 > 1$. As $X_1^{-2e} X_2^{-2e} N^3 > 1$ implies $X_1^{-e} X_2^{-e} N^2 > 1$ all solutions that can be calculated with the former approach can also be calculated with the latter. Hence, it is better only to include the shifts polynomials f_1 and f_2 in the shift polynomial set.
Same results hold for higher degree shift monomials leading to the same dependencies. Thus, multiplication by monomials in other variables does not help to obtain a better bound. This again meets our expectations as the polynomials are independent.

So why did we consider these shared lattices at all? Adding a single equation in both variables to the lattice can significantly improve the bound. To see this, let us return to the system of equations (5.2). We assume that $\bar{x}_1 \le \bar{x}_2$ and, thus, $X_1 \le X_2$. Suppose $p(x_1, x_2)$ contains the monomial x_2^e and only one monomial $x_1^{i_1} x_2^{i_2}$ not occurring in f_1 or f_2. Further, let $i_2 < e$. The idea is to substitute x_2^e by the new monomial and build up the corresponding lattice L. Then more powers of X_1^{-1} but less powers of X_2^{-1} appear in the determinant. To see if this helps to improve our analysis, we apply the simplified condition (2.14) to our construction. That is, we require $|\det(L)| > 1$ and, as $X_1^{-1} > X_2^{-1}$, the bound is improved. It is then better adapted to the different sizes of the unknowns. As substitution becomes more difficult when monomials reappear due to shifts, we do not explicitly perform the substitution but add further columns in the lattice. Let us continue with the example to see how the technique is used. Let $p(x_1, x_2) = x_2^e + p_1 x_1^{e-2} x_2 + p_0$ with $p_1, p_0 \in \mathbb{Z}$. We define a new lattice L_p by a basis matrix which includes p:

$$\mathbf{B_p} = \begin{pmatrix} 1 & 0 & 0 & p_0 & -c_1 & -c_2 \\ 0 & \frac{1}{X_1^e} & 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{X_1^{e-2} X_2} & p_1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & N & 0 \\ 0 & 0 & 0 & 0 & 0 & N \end{pmatrix}. \tag{5.15}$$

Remark that the structure of $\mathbf{B_p}$ is different to what we have seen so far. It combines column vectors used in the modular and in the integer case. If one compares it to $\mathbf{B}$, the column corresponding to the diagonal entry $\frac{1}{X_2^e}$ is replaced by the column corresponding to the new equation $p(x_1, x_2) = 0$. We no longer have to introduce the monomial explicitly in the diagonal matrix but get it "for free" with the help of p. However, as p includes another new monomial, this has to be added on the diagonal.

With unimodular transformations $\mathbf{B_p}$ can be transformed to

$$\begin{pmatrix} 1 & \frac{c_1}{X_1^e} & 0 & 0 & 0 & -c_2 - p_0 \\ 0 & -\frac{N}{X_1^e} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{X_1^{e-2}X_2} & 0 & 0 & -p_1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & \frac{1}{X_1^e} & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & N \end{pmatrix}. \tag{5.16}$$

Without loss of generality we can assume N and p_1 to be coprime. (Otherwise we can compute a divisor of N which compromises the RSA public key (N, e).) Therefore, there is a unimodular transformation of $\mathbf{B_p}$ into

$$\begin{pmatrix} (\mathbf{B_p})_\mathbf{S} & \mathbf{0} \\ * & \mathbf{I_3} \end{pmatrix}. \tag{5.17}$$

Further applying Coppersmith's method, we get an additional equation in the unknowns if the condition $|\det(L_p)| > 1$ is fulfilled. (For a better understanding we again use the simplified condition.) It is $\det(L_p) = X_1^{-2e+2}X_2^{-1}N^2 > 1 \Leftrightarrow X_1^{2e-2}X_2 < N^2$.
Note that the target vector here is $(\mathbf{t_p})_\mathbf{S} = (1, \frac{x_1^e}{X_1^e}, \frac{x_1^{e-2}x_2}{X_1^{e-2}X_2})$. Furthermore, remark that plugging a solution m_1 into $p(x_1, x_2)$ gives an equation in only one unknown x_2 over the integers. The solution of the second unknown m_2 can, thus, be of any size if m_1 is small enough. This is exactly what the determinant condition gives. Assume for example that m_2 is of full size, i. e. $X_2 = N$. The condition then becomes $X_1 < N^{\frac{1}{2e-2}}$.
With respect to our initial problem, we have not improved the bounds this way. If we regard f_1 separately, the upper bound on X_1 is $N^{\frac{1}{e}}$, which is better than $N^{\frac{1}{2e-2}}$. Using $p(m_1, x_2)$, any solutions $m_2 \in \mathbb{Z}_N$ can then be determined. However, using additional shifts, the condition can be improved further. We will show this with regard to a more general setting. The above example should only illustrate the general method. For instance, if $k > 2$, and if we have only one additional polynomial p denoting an implicit relation of the messages $m_1, \ldots, m_k$, the technique might be useful. Namely, we can use p only to determine one message of arbitrary size provided that the other messages are given beforehand. This implies that all other messages have to fulfill the condition $X_i < N^{\frac{1}{e}}$. Building a lattice with the original equations and the implicit relation, however, might allow to calculate different solutions. That is, more than one message may be of size larger than $N^{\frac{1}{e}}$ if there are other messages which are significantly smaller.

In what follows we will give an algorithm to change a lattice basis and apply Coppersmith's algorithm when given a set of k independent equations $f_i(x_i) \equiv 0 \pmod{N}$ of degree δ_i, $i = 1, \ldots, k$, and a set of $\kappa < k$ implicit relations $p_i(x_1, \ldots, x_k) = 0$, $i = 1, \ldots, \kappa$. The major steps of the algorithm are as follows:

1. For each p_i determine the most costly monomial not yet introduced and denote it by m_i.

2. Determine a determinant preserving shift polynomial set $\mathcal{F}$ with shifts of $f_1, \ldots, f_k$ that uses many m_i and powers of them.

3. Construct a standard lattice basis $\mathbf{B}$ with regard to the shift polynomial set $\mathcal{F}$.

4. Substitute columns of $\mathbf{D}$ by columns corresponding to shifts of p_i.

5. Perform Coppersmith's method with the given basis matrix.

We elaborate on the steps of the algorithm.

Determine costly monomials
Recall from the example that the implicit relations are used to eliminate monomials and, if necessary, replace them by others. For each polynomial $p_i(x_1, \ldots, x_k)$ we have to decide which monomial it shall introduce in the lattice. Further, we have to distinguish useful from non-useful replacements.
The contribution of a monomial $m(x_1, \ldots, x_k)$ to $\mathbf{D}$ and, thereby, to the determinant of the lattice is M^{-1}, where $M := m(X_1, \ldots, X_k)$ denotes the evaluation of m on the upper bounds of the unknowns. For the rest of this paragraph when talking about "larger" or "smaller" monomials we think of larger or smaller with respect to the absolute values of the monomials if evaluated at $(X_1, \ldots, X_k)$. Given a single polynomial p_1, it is, therefore, best to choose m_1 as the largest monomial of p_1. When regarding more than one polynomial, we have to take into account that different polynomials might share monomials. For example, the same monomial should not be introduced by more than one polynomial. A useful replacement can then be performed in the following way:
Let $\mathcal{P} := \{p_1, \ldots, p_\kappa\}$ and $\mathcal{M} := \text{Mon}(\mathcal{P})$ denote the set of all monomials in $\mathcal{P}$. For $i = 1, \ldots, \kappa$ perform the following steps:
Define m_i to be the largest monomial in $\mathcal{M}$ and set $\tilde{p}_i$ to be the corresponding polynomial. If there is more than one polynomial in which m_i occurs, then we look at the other monomials in both polynomials. Let p and q denote these two polynomials. Further, let $\mathcal{M}_p := \text{Mon}(p) \setminus \text{Mon}(q)$ and $\mathcal{M}_q := \text{Mon}(q) \setminus \text{Mon}(p)$. Suppose the largest monomial in $\mathcal{M}_p$ is greater than the largest monomial in $\mathcal{M}_q$. Then choose $\tilde{p}_i$ to be the polynomial q. If there are more than two polynomials, compare them successively. This leads to a unique choice of $\tilde{p}_i$ as this comparison is transitive.
Redefine $\mathcal{P} := \mathcal{P} \setminus \{\tilde{p}_i\}$ and $\mathcal{M} := \text{Mon}(\mathcal{P}) \setminus \{m_1, \ldots, m_i\}$.
In the end, we have defined a sequence $m_1, \ldots, m_\kappa$ to be introduced, and ordered the polynomials $\tilde{p}_1, \ldots, \tilde{p}_\kappa$ to correspond to this.

Determine a basic shift polynomial set
Unfortunately, there is no good generic method to determine a shift polynomial set given any set of polynomial equations $f_i(x_1, \ldots, x_k) \equiv 0 \pmod{N}$, $i = 1, \ldots, k$. Most times, better bounds can be achieved if the choice of shift polynomials is adapted to the specific set of polynomial equations and additional relations $p_i(x_1, \ldots, x_k) = 0$, $i = 1, \ldots, \kappa$. Nevertheless, we will give two basic approaches here. Note that we define the shift polynomial set only with respect to the polynomial equations $f_i(x_1, \ldots, x_k) \equiv 0 \pmod{N}$, the implicit

relations are only included after the lattice basis has been constructed. Then they are used to increase the value of the determinant and, hence, improve the bound.
We distinguish two cases with respect to the monomials which occur in the polynomials f_i. Case 1 denotes that the polynomials f_i share most of the variables and many monomials, whereas case 2 denotes that only a few monomials occur in two different polynomials. Furthermore, there are variables only occurring in some of the polynomials. The example of RSA with related messages falls into this category.
In the first case, we define the shift polynomial set as follows. For $\lambda \in \mathbb{N}$ let $\mathcal{M} := \text{Mon}\left((\sum_{i=1}^{k} f_i)^{\lambda}\right)$. The general idea is to follow the construction by Ellen Jochemsz and Alexander May [JM06]. Like them, we assume without loss of generality all polynomials f_i to be monic. Furthermore, we assume that all monomials which occur in $f_i^{j_1}$ also occur in $f_i^{j_2}$ provided that $j_2 \geq j_1$.
For any monomial which shall occur, a polynomial by which it is introduced is defined. When doing so, a polynomial power as large as possible is used. This is because any polynomial power f_i^l in the shift polynomial set contributes a power N^l to the determinant. Recall that the intention is to have a large determinant as a large determinant implies a better bound. For each monomial m we search for a power as large as possible of a leading monomial $LM(f_i)$ which divides m. In contrast to the approach in the case of one polynomial, we have to take into account which of the polynomials f_i to take. For $i = 1, \ldots, k$ and $l = 0, \ldots, \lambda + 1$ we define the set of monomials

$$\mathcal{M}_{il} := \{m \mid m \in \mathcal{M} \text{ and } \frac{m}{LM(f_i)^l} \text{ is monomial of } f_i^{\lambda - l}\}.$$

The union of $\mathcal{M}_{il}$ for equal values of l is denoted by $\mathcal{M}_l := \cup_{i=1}^{k} \mathcal{M}_{il}$. Thus, a monomial m in $\mathcal{M}_l \setminus \mathcal{M}_{l+1}$ can be introduced by a shift polynomial of the form $\frac{m}{LM(f_i)^l} f_i^l$. Now, we define the polynomials

$$g_m(x_1, \ldots, x_k) := \frac{m}{LM(f_i)^l} f_i^l, \quad l = 0, \ldots, \lambda, \quad i = 1, \ldots, k \text{ and}$$

$$m \in \mathcal{M}_{il} \setminus \left(\mathcal{M}_{l+1} \cup \left(\cup_{j=0}^{i-1} M_{jl}\right)\right).$$

We set $\mathcal{F}$ to be the set of all polynomials g_m.

In the second case, the joint shift polynomial set $\mathcal{F}$ is defined by joining the shift polynomial sets $\mathcal{F}_i$ of the polynomials f_i shifted individually, exactly following the standard approach by Ellen Jochemsz and Alexander May [JM06]. For a general system of equations, both approaches can be combined in an appropriate manner.

Construct a lattice basis
Given a shift polynomial set, the construction of a lattice basis is straightforward and can be performed as in Section 2.3.

Substitute columns of the basis
For each $m_i^{\lambda_i}$, $i \in \{1, \ldots, \kappa\}$, and for all entries m in $\mathbf{D}$, check if m_i divides the monomial m. If yes, check if $\{\frac{m}{m_i^{\lambda_i}} p_i(x_1, \ldots, x_k)\} \cup \mathcal{F}$ is still determinant preserving. If yes, then substitute the column of $\mathbf{D}$ by the coefficient vector of $\frac{m}{m_i^{\lambda_i}} p_i(x_1, \ldots, x_k)$. If necessary, introduce further monomials occurring in $\frac{m}{m_i^{\lambda_i}} p_i(x_1, \ldots, x_k)$ on the diagonal. That is, add another row and column for each new monomial. As the monomials m_i are ordered such that their size decreases, substitutions in monomials occurring only due to other substitutions are covered by this approach.

Perform Coppersmith's method
Having constructed a lattice basis $\mathbf{B}$, we can now proceed as described in Section 2.3. The upper bounds on the sizes of the unknowns can be determined with the folklore analysis and depend on the determinant of $\mathbf{B}$.

The algorithm described here gives a method to analyze any modular multivariate system of equations with the same modulus and additional integer relations. The method used to introduce new columns is a variant of the unraveled linearization technique presented by Mathias Herrmann and Alexander May in [HM09] to analyze pseudo random number generators. In unraveled linearization new monomials are directly substituted, whereas we introduce new polynomials in the lattice. For integer relations, however, this does not make a difference. In case of modular relations all sharing the same modulus, unraveled linearization can be performed analogously, whereas there is no obvious way to include the modular relations in the lattice without introducing a new monomial. However, if powers of modular polynomials occur in the shift polynomial set, direct substitutions are no longer possible with unraveled linearization either. The method of unraveled linearization, too, is adapted to the special shape of the polynomials and the implicit relations. Unfortunately, this implies that the algorithm in this form cannot be automated. In practical analyses human interaction is required.

5.2.2 RSA with Random Paddings

Another example of a system of equations with a common modulus can be derived from RSA with random paddings. Let $(N, 3)$ be a user's public key, where N is an n-bit number. A message m of $\alpha < n$ bits is then encrypted as $c \equiv \left(2^{\alpha+\tau} v + 2^\tau m + w\right)^3 \pmod{N}$ for a random τ-bit number w and a random $(n - \alpha - \tau)$-bit number v. The message m as well as the values v and w are secret and not known to any attacker. At first sight this problem seems to be easier than the example given in the previous subsection as the relations between two different RSA encrypted messages are explicit, linear and monic. The relations, however, are no longer completely known but include further (relatively small) unknowns.
The case of two such encryptions of the same message with different paddings is described

and analyzed in [Jut98]. Let

$$\begin{aligned} f_1(m) := \left(2^{\alpha+\tau}v_1 + 2^\tau m + w_1\right)^3 - c_1 &\equiv 0 \pmod N \\ \text{and} \quad f_2(m) := \left(2^{\alpha+\tau}v_2 + 2^\tau m + w_2\right)^3 - c_2 &\equiv 0 \pmod N \end{aligned}$$

be the polynomial equations derived from two different encryptions of the same message m with paddings (v_1, w_1) and (v_2, w_2). Set

$$\begin{aligned} m_1 &= 2^{\alpha+\tau}v_1 + 2^\tau m + w_1 \\ \text{and} \quad m_2 &= 2^{\alpha+\tau}v_2 + 2^\tau m + w_2\,. \end{aligned}$$

Charanjit S. Jutla aims at using the technique applied by Don Coppersmith et al. [CFPR96] described in Section 3.1 to solve RSA with affinely related messages and known relations. In contrast to the example given there, however, in this case we have further unknowns from the unknown paddings. Therefore, in order to determine these unknowns, set $v_{12} := v_2 - v_1$, $w_{12} := w_2 - w_1$ and $\Delta_{12} := 2^{\alpha+\tau}v_{12} + w_{12}$. With this notation it follows that $m_2 \equiv m_1 + \Delta_{12} \pmod N$. To define a system of equations, we replace the values by variables. Let x represent m_1, t_i represent v_i, u_i represent w_i and $\rho_{12} := 2^{\alpha+\tau}t_{12} + u_{12} = 2^{\alpha+\tau}(t_2 - t_1) + (u_2 - u_1)$ correspond to Δ_{12}.
Thus, we regard the following system of equations

$$\begin{aligned} g_1(x) := x^3 - c_1 &\equiv 0 \pmod N \\ \text{and } g_{12}(x, \rho_{12}) := (x + \rho_{12})^3 - c_2 &\equiv 0 \pmod N\,. \end{aligned}$$

To eliminate the unknown x, Charanjit S. Jutla computes the resultant of the two polynomials with respect to x in $\mathbb{Z}_N$, namely $\mathrm{res}_{12}(\rho_{12}) := \mathrm{Res}_x(g_1(x), g_{12}(x, \rho_{12}))$. Resubstituting ρ_{12}, one gets a bivariate polynomial r_{12} in (t_{12}, u_{12}) with small roots $v_2 - v_1$ and $w_2 - w_1$ modulo N. Applying Coppersmith's method to $r_{12}(t_{12}, u_{12})$, the roots can be found as long as the unknowns are small enough. The product of the unknowns ought to be bounded by $N^{\frac{1}{9}}$, i. e. the total padding must be smaller than one ninth of the bitsize of N. This result can be obtained by applying a generalization of Coppersmith's original method and using a generalized lower triangle to build up the shift polynomial set [JM06]. Subsequently, we can substitute ρ_{12} by the solution $\Delta_{12} := 2^{\alpha+\tau}(v_2 - v_1) + (w_2 - w_1)$ in g_{12}. Then $g_1(x) \equiv 0 \pmod N$ and $g_{12}(x, \Delta_{12}) \equiv 0 \pmod N$ denote two equations derived from RSA encryption with known relation. Such a system can be solved by the method of Don Coppersmith, Matthew Franklin, Jacques Patarin and Michael Reiter [CFPR96] presented in Section 3.1.
Now let us consider what happens if we get another encryption of the same message with a different padding. Intuitively, an additional equation implies additional information and the bound should improve. This is trivially the case as we no longer have to require that $|(v_2 - v_1)(w_2 - w_1)|$ is smaller than $N^{\frac{1}{9}}$. Let

$$f_3(m) := \left(2^{\alpha+\tau}v_3 + 2^\tau m + w_3\right)^3 - c_3 \equiv 0 \pmod N$$

denote the equation derived from new encryption of the message m. Then, analogously to the combination of f_1 and f_2 presented above, we can also combine f_3 with either f_1 or f_2.

We obtain the analogue conditions $|(v_3 - v_i)(w_3 - w_i)| < N^{\frac{1}{9}}$, $i = 1, 2$. Thus, if any of the products of the unknowns is smaller than $N^{\frac{1}{9}}$, the message m can be determined. That is, we require the following condition:

$$\min\{|(v_2 - v_1)(w_2 - w_1)|, |(\underbrace{v_3 - v_1}_{=:v_{13}})(\underbrace{w_3 - w_1}_{=:w_{13}})|, |(\underbrace{v_3 - v_2}_{=:v_{23}})(\underbrace{w_3 - w_2}_{=:w_{23}})|\} < N^{\frac{1}{9}} .$$

The question now is if we can improve on this result using lattice-based techniques like Coppersmith's method. Unfortunately, there are strong arguments that this is not the case. First of all, we combine f_1 and f_3 in the same manner as we have combined f_1 and f_2 to get a polynomial $g_{13}(x) := (x + \rho_{13})^3 - c_3$. Here $\rho_{13} := 2^{\alpha+\tau}(\underbrace{t_3 - t_1}_{=:t_{13}}) + (\underbrace{u_3 - u_1}_{=:u_{13}})$. The variables t_i and u_i again correspond to the target values v_i and w_i, respectively. We would like to eliminate the variable x as it corresponds to a large value and compute the resultant of g_1 and g_{13} to get a polynomial $\text{res}_{13}(\rho_{13}) := \text{Res}_x(g_1(x), g_{13}(x, \rho_{13}))$. By resubstituting ρ_{13}, we get the bivariate polynomial $r_{13}(t_{13}, u_{13})$ of maximum total degree nine. Note that the variables t_{12} and t_{13} as well as the variables u_{12} and u_{13} are pairwise independent as they are derived from independent variables. Consequently, the bounds we obtain for a joint shift polynomial set will not improve on the bound we obtain for separate shift polynomial sets. Explanations for this have been given in Subsection 5.2.1 for a system of univariate equations in independent variables. The argumentation given in the proof of Theorem 5.2.3 analogously works in case of more than one variable. Thus, we get the condition

$$|(v_2 - v_1)(w_2 - w_1)(v_3 - v_1)(w_3 - w_1)| \quad < \quad N^{\frac{2}{9}} .$$

This implies

$$\min\{|(v_2 - v_1)(w_2 - w_1)|, |(v_3 - v_1)(w_3 - w_1)|\} \quad < \quad N^{\frac{1}{9}} ,$$

the condition we have already obtained using the trivial analysis.

So far we have combined g_1 and g_{12} as well as g_1 and g_{13}. A third alternative is to combine g_{12} and g_{13}. Note that $\text{Res}_x(g_{12}(x, \rho_{12}), g_{13}(x, \rho_{13}))$ is a polynomial in $\rho_{13} - \rho_{12}$ of maximum degree nine. Resubstituting both variables results in the polynomial $r_{23}((\underbrace{t_3 - t_2}_{=:t_{23}}), (\underbrace{u_3 - u_2}_{=:u_{23}}))$ likewise of maximum total degree nine. If t_{23} and u_{23} were independent of t_{12}, t_{13}, u_{12} and u_{13}, we could again follow the argumentation given for two equations and the bound would not improve. The variables t_{23} and u_{23}, however, depend on t_{12}, t_{13}, u_{12} and u_{13} in a trivial way: It is $t_{23} = t_3 - t_2 = t_3 - t_1 + t_1 - t_2 = t_{13} - t_{12}$ and, analogously $u_{23} = u_{13} - u_{12}$. Thus, $\rho_{23} = \rho_{13} - \rho_{12}$. These are integer relations we might again use to eliminate variables in the construction of the lattice basis . We define the polynomial $r(\rho_{12}, \rho_{13}, \rho_{23}) := -\rho_{23} + \rho_{13} - \rho_{12}$ corresponding to the relation $-\rho_{23} + \rho_{13} - \rho_{12} = 0$.

The question is if this relation helps to improve the bound. Unfortunately, we can give strong arguments that it does not. To see why this is the case, we look at possible lattice constructions. We pursue two different approaches to choose a shift polynomial set. Then we argue why these shift polynomial sets do not help to improve the bound. As the lattice constructions we consider are relatively general, we can exclude many at first sight promising shift polynomial sets. Note however, that we do not prove that no better lattice construction exists.

The first shift polynomial set is based on shifts of $r_{12}(\rho_{12}), r_{13}(\rho_{13})$ and $r_{23}(\rho_{23})$. Then, like in the previous section, we use the known relation r between the unknowns to introduce some of the monomials which occur in the shift polynomial set. As usual, let $\mathbf{D}$ denote the left part of the basis matrix we use in Coppersmith's algorithm. Adding a multiple of the relation r in the shift polynomial set only influences the determinant of the lattice by changing the set of monomials which have to be introduced. That is, the elements on the diagonal of $\mathbf{D}$ are changed. This implies that the bound can only be improved if more or larger monomials can be eliminated from $\mathbf{D}$ than have to be introduced. However, this is not the case as we show by a counting argument.
The second shift polynomial set is constructed with respect to the polynomials $r_{12}(\rho_{12})$, $r_{13}(\rho_{13})$ and $r_{23}(\rho_{12}, \rho_{13})$. That is, we use the known relation to describe r_{23} directly as a polynomial in the unknowns ρ_{12} and ρ_{13}.
We determine the maximum total degree of a monomial in the shift polynomial set. We assume all monomials with degree $l = 3\lambda, \lambda \in \mathbb{N}$, of small enough degree to occur in the shift polynomial set. This is a sensible assumption due to the structure of r_{23}.
Then we calculate the maximum number $\max_r$ of shifts of each polynomial that can be included in the shift polynomial set. By this, we maximize the number of factors N^l we can obtain in the determinant of our lattice.
On the other hand, we determine the maximum number $\max_d$ of polynomials that form a determinant preserving set. An upper bound for this is given by the number of monomials that occur in $\mathcal{F}$. Then we compare the number of potential shift polynomials to the upper bound on the size of a determinant preserving shift polynomial set. With growing maximum total degree, we observe that $\max_d$ and $\max_r$ are of approximately the same size. More precisely, $\max_d - \max_r = o(l^2)$, whereas $\max_d = \Omega(l^2)$ and $\max_r = \Omega(l^2)$. Thus, if we take all possible shifts of one of the polynomials, we cannot add any further shifts to our shift polynomial set. Otherwise, the set would not remain determinant preserving. Consequently, we may assume that adding shifts of a second polynomial does not help to improve the bound. Shifts of a second polynomial can only be included if shifts of the first polynomial are left out.
We will now describe the two variants in detail.

Approach 1
The first variant is to build up the lattice as before. Then we can add shifts of the integer polynomial $r(\rho_{12}, \rho_{13}, \rho_{23}) := -\rho_{23} + \rho_{13} - \rho_{12}$. By this additional relation we can eliminate monomials as in the previous section. During the elimination process, however, we introduce more new monomials than we can eliminate old ones without getting further positive factors in the determinant. Let us have a look at a simple basis matrix constructed

with the shift polynomial set $\{r_{12}, r_{13}, r_{23}\}$. Furthermore, let us assume $v_i = 0$ for $i = 1, 2, 3$. Then we can omit the variables t_{ij}. Instead, we can work with $\rho_{ij} = u_{ij}$. We set

$$\begin{array}{c|cccc|}
 & & r_{12} & r_{13} & r_{23} \\
1 & & (c_1-c_2)^3 & (c_1-c_3)^3 & (c_2-c_3)^3 \\
\rho_{12}^3 & & 3c_1^2+21c_1c_2+3c_2^2 & & \\
\rho_{12}^6 & & 3(c_1-c_2) & & \\
\rho_{12}^9 & & 1 & & \\
\rho_{13}^3 & & & 3c_1^2+21c_1c_3+3c_3^2 & \\
\rho_{13}^6 & \mathbf{D} & & 3(c_1-c_3) & \\
\rho_{13}^9 & & & 1 & \\
\rho_{23}^3 & & & & 3c_2^2+21c_2c_3+3c_3^2 \\
\rho_{23}^6 & & & & 3(c_2-c_3) \\
\rho_{23}^9 & & & & 1 \\
 & & N & & \\
 & & & N & \\
 & & & & N
\end{array} =: \mathbf{B}\,. \qquad (5.18)$$

Let R_{ij} denote an upper bound on the absolute value of ρ_{ij}. The matrix

$$\mathbf{D} := \mathrm{Diag}(1, R_{12}^{-3}, R_{12}^{-6}, R_{12}^{-9}, R_{13}^{-3}, R_{13}^{-6}, R_{13}^{-9}, R_{23}^{-3}, R_{23}^{-6}, R_{23}^{-9})$$

is the diagonal matrix corresponding to the monomials that occur in the shift polynomial set. Now we would like to use the relation r to reduce the number of entries on the diagonal of $\mathbf{D}$. We choose to eliminate ρ_{23}. As the problem is symmetric in ρ_{12}, ρ_{13} and ρ_{23} (disrespecting signs) the argumentation is the same for any other variable. The relation r is linear, the monomials which occur in the shift polynomial set are of degrees $3, 6$ and 9. Hence, in order to eliminate powers of ρ_{23}, we have to multiply r by further monomials. To eliminate ρ_{23}^3, we use the shift polynomial $\rho_{23}^2 r(\rho_{12}, \rho_{13}, \rho_{23}) = -\rho_{23}^3 + \rho_{13}\rho_{23}^2 - \rho_{12}\rho_{23}^2$. This shift polynomial introduces two new monomials $\rho_{13}\rho_{23}^2$ and $\rho_{12}\rho_{23}^2$. Recall that a useful new set of shift polynomials should enlarge the size of the determinant. The contribution of each monomial to the determinant is the inverse of the monomial evaluated at the upper bounds of the unknowns. This implies that the factor R_{23}^{-3} in the determinant is replaced by a factor of $R_{12}^{-1}R_{13}^{-1}R_{23}^{-4}$. As a single replacement this cannot be useful as $R_{12}^{-1}R_{13}^{-1}R_{23}^{-4} < R_{23}^{-3}$. We can do better by using further shifts of the polynomial r. That is, we add the set $\{\rho_{23}^2 r, \rho_{23}\rho_{13} r, \rho_{23}\rho_{12} r, \rho_{13}^2 r, \rho_{13}\rho_{12} r, \rho_{12}^2 r\}$ to the shift polynomial set. By adequately ordering the six polynomials, we can use them to introduce six monomials. Note that the monomials ρ_{12}^3 and ρ_{13}^3 already exist in the lattice. However, there are ten monomials of degree 3 in ρ_{12}, ρ_{13} and ρ_{23}, namely, $\rho_{12}^{i_{12}}\rho_{13}^{i_{13}}\rho_{23}^{i_{23}}$ such that $i_{12} = 0, \ldots, 3\,, i_{13} = 0, \ldots, 3 - i_{12}$ and $i_{23} = 3 - i_{12} - i_{13}$. Thus, we obtain two new monomials.

In the case of higher degree monomials, similar substitutions can be made. However, the difference between the number of monomials which can be introduced by a polynomial in the shift polynomial set and the total number of monomials of this degree which we get increases with the degree. Thus, we have to introduce even more new monomials. Suppose that we would like to substitute ρ_{23}^l for an integer l which is divisible by 3. Only introducing the monomial by the shift $\rho_{23}^{l-1}r$ results in two new monomials $\rho_{23}^{l-1}\rho_{13}$ and $\rho_{23}^{l-1}\rho_{12}$. Hence, for the determinant to become larger, we need $R_{23}^{-2(l-1)}R_{12}^{-1}R_{13}^{-1} > R_{23}^{-l} \Leftrightarrow 1 > R_{23}^{l-2}R_{12}R_{13}$, which is never fulfilled. Successively adding more shifts leads to a similar condition. Each shift polynomial used to introduce one already existing monomial also contains another monomial not yet contained in the set of monomials. Thus, when eliminating one value from the determinant, we have to introduce a different one.

Alternatively, we can introduce all possible monomials by shifts of r. Altogether, there are $\sum_{i=0}^{l}\sum_{j=0}^{l-i} 1 = \frac{1}{2}l^2 + \frac{3}{2}l + 1$ monomials of total degree l in ρ_{12}, ρ_{13} and ρ_{23}. The monomials ρ_{12}^l and ρ_{13}^l have already been introduced by shifts of r_{12} and r_{13}, respectively. Any other monomial has to be introduced by a shift of r. The number of shifts of r to give monomials of total degree l is limited by the number $\frac{1}{2}l^2 + \frac{1}{2}l$ of monomials of total degree $l-1$ with which we can shift r. Consequently, in total $\left(\frac{1}{2}l^2 + \frac{3}{2}l + 1\right) - \left(\frac{1}{2}l^2 + \frac{1}{2}l\right) - 2 = l - 1$ monomials of degree l have to be introduced anew. A basis matrix corresponding to this is given by

$$\begin{array}{c} \\ 1 \\ \rho_{12}^3 \\ \rho_{12}^6 \\ \rho_{12}^9 \\ \rho_{12}^2\rho_{13} \\ \vdots \\ \rho_{12}\rho_{13}^8 \\ \rho_{13}^3 \\ \rho_{13}^6 \\ \rho_{13}^9 \\ \rho_{12}^2\rho_{23} \\ \vdots \\ \rho_{13}\rho_{23}^8 \\ \rho_{23}^3 \\ \rho_{23}^6 \\ \rho_{23}^9 \\ \\ \\ \\ \end{array} \begin{array}{c} \\ \left(\begin{array}{cccc} & (c_1-c_2)^3 & (c_1-c_3)^3 & (c_2-c_3)^3 \\ & 3c_1^2+21c_1c_2+3c_2^2 & & \\ & 3(c_1-c_2) & & \\ & 1 & & \\ & & & \\ & & \mathbf{0} & \\ & & & \\ & & 3c_1^2+21c_1c_3+3c_3^2 & \\ \tilde{\mathbf{D}} \quad \mathbf{R} & & 3(c_1-c_3) & \\ & & 1 & \\ & & & \\ & & \mathbf{0} & \\ & & & \\ & & & 3c_2^2+21c_2c_3+3c_3^2 \\ & & & 3(c_2-c_3) \\ & & & 1 \\ & N & & \\ & & N & \\ & & & N \end{array}\right) \end{array} =: \tilde{\mathbf{B}}\,.$$

with the columns headed r_{12}, r_{13}, r_{23} respectively.

The matrix

$$\tilde{\mathbf{D}} := \mathrm{Diag}(1, R_{12}^{-3}, R_{12}^{-6}, R_{12}^{-9}, R_{12}^{-2}R_{13}^{-1}, R_{12}^{-1}R_{13}^{-2}, R_{12}^{-5}R_{13}, \ldots, R_{12}^{-1}R_{13}^{-8}, R_{13}^{-3}, R_{13}^{-6}, R_{13}^{-9})$$

is the diagonal matrix corresponding to the monomials that occur in the new shift polynomial set. The matrix $\mathbf{R}$ is the matrix induced by the coefficient vectors of the polynomials

$\rho_{12}^2 r, \rho_{12}\rho_{13}r, \dots, \rho_{23}^2 r, \rho_{12}^5 r, \dots, \rho_{23}^5 r, \rho_{12}^8 r, \dots, \rho_{23}^8 r$. Note that $\mathbf{R} = \begin{pmatrix} \mathbf{R_u} \\ \mathbf{R_l} \end{pmatrix}$, where $\mathbf{R_l}$ is an upper triangular matrix with value one on its diagonal. That is because the diagonal entries correspond to monomials introduced by the shifts of r. Due to this structure, it is $\det(\mathbf{R_l}) = 1$. Thus, $\det(\tilde{\mathbf{B}}) = \det(\tilde{\mathbf{D}})\det(\mathbf{R_l})N^3 = \det(\tilde{\mathbf{D}})N^3$.
In order for the substitution to give a better bound, the new determinant ought to be larger than the previous one. The condition for this is $\det(\tilde{\mathbf{D}}) > \det(\mathbf{D})$. This is equivalent to $R_{23}^{18} > R_{12}^{54}R_{13}^{54} \Leftrightarrow R_{23} > R_{12}^3 R_{13}^3$. Asymptotically, the number of monomials which have to be introduced increases and the condition gets even worse. The reason for this can be seen easily. Let ρ_{23}^l for some $l \in \mathbb{N}$ denote the monomial that shall be substituted. Then, as argued above, $l-1$ monomials of degree l have to be newly introduced. Multiplying these monomials gives a monomial of total degree $l^2 - l$. Thus, the determinant gets larger provided that $R_{23}^{-l} < R_{12}^{-l_{12}}R_{13}^{-l_{13}}R_{23}^{-l_{23}}$ with $l_{12} + l_{13} + l_{23} = l^2 - l$. If $l_{23} \geq l$, this condition cannot be fulfilled. Hence, $l_{23} < l$. Then the condition is equivalent to $R_{23}^{l-l_{23}} > R_{12}^{l_{12}}R_{13}^{l_{13}}$. It is $l_{12} + l_{13} = l^2 - l - l_{23} \overset{l>2}{\geq} 2l - l_{23} > 2(l - l_{23})$. Further, l_{12} and l_{13} are greater than $l - l_{23}$ each. This can be seen as follows. Assume the contrary, i.e. $l_{13} \leq l - l_{23}$. (The case of $l_{12} \leq l - l_{23}$ can be treated analogously.) Then, on the one hand, $l_{12} = l^2 - l - l_{23} - l_{13} \geq l^2 - 2l$. On the other hand, it is $l^2 - l - l_{23} = l(l-2) + l - l_{23}$. Consequently, the product we consider consists of at least $l-2$ monomials not comprising any power of ρ_{23}. If we take the highest powers of ρ_{12} possible in these monomials, they sum up to $\sum_{i=3}^{l} i = \frac{1}{2}l^2 + \frac{1}{2}l - 3$. To get the highest power in ρ_{12} we assume the last two monomials to be $\rho_{12}^{l-1}\rho_{23}$ and $\rho_{12}^{l-2}\rho_{13}\rho_{23}$. This implies $l_{12} < \frac{1}{2}l^2 + \frac{5}{2}l - 6$. Combining the two conditions, we get $l^2 - 2l \leq l_{12} < \frac{1}{2}l^2 + \frac{5}{2}l - 6$. This can only be fulfilled if $l < 8$. Hence, if the value of l increases, the influence of monomials of degree 3 and 6 on the size of the determinant will be compensated for by higher degree monomials. That is to say, the substitution does not help to improve the bound we can obtain asymptotically.
Consequently, any of these conditions can only be fulfilled provided that the weaker condition $R_{23} > R_{12}R_{13}$ holds as well. Note, however, that $|\rho_{23}| = |\rho_{12} - \rho_{13}| \leq |\rho_{12}| + |\rho_{13}| \leq 2\max\{|\rho_{12}|, |\rho_{13}|\} \overset{\log(N)\geq 512}{\leq} |\rho_{12}\rho_{13}|$. Thus, an upper bound R_{23} such that $R_{23} > R_{12}R_{13}$ would not fit to the problem. Consequently, this approach should not be pursued.

Approach 2
As a second variant, we can directly substitute $\rho_{23} = \rho_{13} - \rho_{12}$ in r_{23}. For notational convenience, we denote by r_{23} the original polynomial in t_{23} and u_{23}, whereas we denote by $\tilde{r}_{23}$ the polynomial $r_{23}(t_{13} - t_{12}, u_{13} - u_{12})$ as polynomial in t_{13}, t_{12}, u_{13} and u_{12}. Then we construct a lattice with respect to r_{12}, r_{13} and $\tilde{r}_{23}$. Remark that all monomials which occur in $\tilde{r}_{23}^l$ for an $l \in \mathbb{N}$ already include all monomials which occur in r_{12}^l or r_{13}^l. Hence, to get a good bound, we can take the folklore shifts of $\tilde{r}_{23}$ and add as many shifts of the two other polynomials as possible. These additional shifts only help to increase the bound without any additional costs. However, we can only take shifts such that the shift polynomial set remains determinant preserving.
Note that as long as we take all possible shifts of $\tilde{r}_{23}$ and the same shifts of $r_{1\iota}$ for a fixed

$\iota \in \{1, 2\}$, we will obtain the original condition $R_{1\iota}R_{23} \leq N^{\frac{2}{9}}$. Thus, for this approach to work, we have to be able to add most of the shifts of $r_{1\iota}$ and a number of shifts of $r_{1(3-\iota)}$ which does not disappear asymptotically. Unfortunately, there is no such determinant preserving shift polynomial set. This can be seen as follows. Of a fixed degree δ, there are exactly $\delta + 1$ monomials in the variables ρ_{12} and ρ_{13}. Thus, the number of different monomials which occur if we include $\tilde{r}_{23}$ up to the power of $l \in \mathbb{N}$ (and shifts of lower powers of $\tilde{r}_{23}$) in our shift polynomial set is $\sum_{k=0}^{3l} (3k + 1) = \frac{27}{2}l^2 + \frac{15}{2}l + 1$. The analogous shifts of r_{12} and r_{13} do not introduce further monomials. The coefficient vectors of the shift polynomials are, thus, vectors in $\mathbb{Z}^{\frac{27}{2}l^2+\frac{15}{2}l+1}$ (if we ignore the moduli).
We look at how many shift polynomials we can maximally include in our shift monomial set. To start, we determine the number of shift polynomials of $\tilde{r}_{23}$. The polynomial $\tilde{r}_{23}^{l-1}$ is shifted by all monomials of degree 0, 3, 6 and all but one of degree 9. Any other polynomial $\tilde{r}_{23}^{i}$ with $i = 1, \ldots, l - 2$ is also shifted by these monomials. Further, it is shifted by all monomials of degree $12, 15, \ldots, 9(l - i)$ which are not divisible by the leading monomial (e. g. ρ_{12}^9 if using graded lexicographical ordering with $\rho_{12} > \rho_{13}$). There are nine such monomials of each degree. The number of the polynomials in the shift set is, thus,

$$\sum_{i=1}^{l-1} (21 + (i - 1)27) = \frac{27}{2}l^2 - \frac{39}{2}l + 6\,.$$

Let us assume we could add the same shifts of r_{12}, r_{13}. Then the number of shift polynomials is $3\left(\frac{27}{2}l^2 - \frac{39}{2}l + 6\right)$. The coefficient vectors of these shift polynomials form a set of $3\left(\frac{27}{2}l^2 - \frac{39}{2}l + 6\right)$ vectors in $\mathbb{Z}^{\frac{27}{2}l^2+\frac{15}{2}l+1}$ (again ignoring the moduli). Hence, at least

$$n_d := 3\left(\frac{27}{2}l^2 - \frac{39}{2}l + 6\right) - \left(\frac{27}{2}l^2 + \frac{15}{2}l + 1\right) = 27l^2 - 66l + 17$$

vectors linearly depend on the others provided that this value is positive. Returning to the polynomials, n_d polynomials are linearly dependent of the set of the other polynomials modulo N and cannot be included in our lattice construction.
On condition that $l > 5$ it is

$$27l^2 - 66l + 17 > \frac{27}{2}l^2 + \frac{15}{2}l + 1\,.$$

This implies that we cannot include any of the shifts of e. g. r_{13} in the shift polynomial set. Hence, the bound cannot get better than the original one. Note that this condition does not state anything about which shifts we can take. It might be that we can include shifts of r_{12} as well as of r_{13}. Their total number, however, may not be larger than the total number of shifts which could be taken of one of the polynomials. Thus, the contribution of powers of N on the determinant will not increase. Then the only way to improve the bound is by using significantly less monomials.
In case of $l = 3$ we cannot exclude 69 but only 62 shift polynomials by these rough estimations. In case of $l = 2$ we cannot even exclude any of the shift polynomials this way.

Actual computations, however, give 5 polynomials which have to be excluded. The bounds obtained with the lattices constructed in the case of $l = 2$, however, are worse than the bound of $N^{\frac{1}{9}}$.
Thus, we return to the former approach. That is, we use the polynomials r_{12}, r_{13} and r_{23} with their basic shifts to construct a lattice basis to use with Coppersmith's method. This way, we can determine Δ_{ij} and, thereby, also m, if $|(v_j - v_i)(w_j - w_i)| < N^{\frac{1}{9}}$ for any pair i,j.

The analysis can be generalized to an arbitrary number of equations in a straightforward manner. The condition we obtain is the same. That is, the difference of two paddings has to be smaller than $N^{\frac{1}{9}}$ for at least one pair of paddings.

In the case of RSA encryption with a different public exponent e, the value $N^{\frac{1}{9}}$ has to be substituted by $N^{\frac{1}{e^2}}$ in all steps of the analysis. The bounds get worse. The running time of the algorithm depends on the running time of Coppersmith's method and the number of possible choices of pairs. Both running times are polynomial in e. Thus, for constant values of e the attack is feasible if $|(v_i - v_j)(w_i - w_j)| < N^{\frac{1}{e^2}}$.
We have seen in our example that combining equations does not help to improve the bound. The reason for this is the independence of the variables ρ_{ij} we can include in one lattice. The main variable occurring in all of the equations is the variable x. It corresponds to the unknown message m. In our analysis, we have eliminated x in the first step by computing resultants with respect to x. If we do not do this, but directly perform Coppersmith's algorithm on one of the original equations, we obtain rather small upper bounds M such that we can determine all m such that $|m| < M$. In this approach taking more than one of the original polynomials and shifting them to obtain shared monomials would allow for a larger value of M. However, the bounds we obtain by any of those analyses are still too small as in practice m has nearly full size. Thus, the above approach in which m is eliminated beforehand is better suited to the problem.
Recapitulating the examples of systems of modular equations with the same modulus, we observe the following: Additional equations in the same variables help to improve certain bounds in case of small lattices provided that the additional equations help to reduce the number of unknown monomials. By additional integer equations the bounds can also be improved. In asymptotically large systems, however, care has to be taken not to introduce too many shifts and, thus, destroy the property that the given system is determinant preserving. Unfortunately, the choice of shift polynomials strongly depends on the original system of equations. Thus, we cannot give a general strategy and bound working for an arbitrary system.
For some specific systems of equations like the ones derived from RSA with random paddings, there are even indications that the known bounds cannot be improved. A reason for this may be the fact that we try to reuse additional information twice. Recall that in a first step, we have computed the resultant of two polynomials. That is, we have included additional information in the new polynomial. In a second step, we aim at getting advantage by combining two such polynomials. However, we have given arguments that this

does not help further. Unfortunately, these arguments can only be given with respect to specific systems of equations. Thus, human interaction is required in these analyses.

5.3 Systems of Equations with Coprime Moduli

Let us again deal with systems of modular multivariate equations. However, in contrast to the previous section we now assume the moduli to be coprime, that is, we analyze *Systems of Modular Multivariate Polynomial Equations with Mutually Coprime Moduli* (SMMPE2-problem).

Definition 5.3.1 (SMMPE2-problem)
Let $k \in \mathbb{N}$, $\delta_1, \ldots, \delta_k \in \mathbb{N}$, and $N_1, \ldots, N_k \in \mathbb{N}$ mutually coprime with $N_1 < \ldots < N_k$. Suppose that $f_i(x_1, \ldots, x_l)$ are polynomials of degree δ_i in $\mathbb{Z}_{N_i}[x_1, \ldots, x_l]$, $i = 1, \ldots, k$, respectively. Let

$$\begin{array}{rcll} f_1(x_1, \ldots, x_l) & \equiv & 0 & \pmod{N_1} \\ f_2(x_1, \ldots, x_l) & \equiv & 0 & \pmod{N_2} \\ & \vdots & & \\ f_k(x_1, \ldots, x_l) & \equiv & 0 & \pmod{N_k} \end{array} \tag{5.19}$$

be a system of multivariate polynomial equations.

Let $X_i < N_i$, $X_i \in \mathbb{R}$, $i = 1, \ldots, l$. Find all common roots $(\bar{x}_1, \ldots, \bar{x}_l)$ of (5.19) with size $|\bar{x}_i| \leq X_i$.

The analysis of system (5.19) is similar to the analysis of system (5.1) with one shared modulus. When considering linear independence, however, we have to take the different moduli into account.
Let $\tilde{\mathcal{F}}$ be the set of shift polynomials used to apply Coppersmith's method (Theorem 2.3.9). Then all $(f, N) \in \tilde{\mathcal{F}}$ consist of a polynomial f of the form $m \prod_{i=1}^{k} f_i^{\lambda_i}(x_1, \ldots, x_l)$ with a monomial $m \in \mathbb{Z}_N[x_1, \ldots, x_l]$, $\lambda_i \in \mathbb{N}_0$, and a modulus $N = \prod_{i=1}^{k} N_i^{\lambda_i} \in \mathbb{N}$. The solution we are searching for is a solution of $f(x_1, \ldots, x_l) \equiv 0 \pmod{N}$ for all $(f, N) \in \tilde{\mathcal{F}}$.
Using these polynomials, a basis matrix is constructed. We again use the notation as with systems with the same modulus. The last columns of the basis matrix corresponding to the shift polynomials are denoted by $\mathbf{F}$. It consists of an upper part $\mathbf{F_c}$ corresponding to the coefficients in the polynomials and a lower part $\mathbf{F_m}$ corresponding to the moduli. The matrix $\mathbf{F_m}$ again forms a diagonal matrix. The values on its diagonal are of type $\prod_{i=1}^{k} N_i^{\lambda_i}$ with $\lambda_i \in \mathbb{N}_0$. We call them diagonal values.
To obtain linear dependence of column vectors of $\mathbf{F}$ modulo an integer $a > 1$, we again have to eliminate the values occurring in $\mathbf{F_m}$ as they only occur in one column. Thus, they cannot be eliminated by linear combination of columns but have to be eliminated by the modular operation.
In contrast to the system in the previous section, however, the diagonal values do not

necessarily share a common divisor. Therefore, we have to check for linear independence modulo a for all $a \neq 1$ dividing more than one of the diagonal values. Namely, we have to check for linear independence modulo the N_i and their divisors. As the N_i are coprime by definition this implies that for a fixed value of j we only have to check the set of columns corresponding to polynomials with $\lambda_j > 0$ for linear independence modulo N_j and its divisors.

Lemma 5.3.2
Let $\tilde{\mathcal{F}} \subset \mathbb{Z}[x_1, \ldots, x_l] \times \mathbb{N}$ be an ordered set of polynomials and corresponding moduli which is not determinant preserving. Let $\mathcal{F}$ be the integer polynomial set induced by $\tilde{\mathcal{F}}$. Let $\mathbf{F}$ be the matrix induced by $\mathcal{F}$ as before. Then there are $\bar{f} \in \mathcal{F}$ and $c_f \in \mathbb{Z}$ such that $\bar{f} \equiv \sum_{f \in \mathcal{F} \setminus \{\bar{f}\}} c_f f \pmod{a}$ where $a|N$ for some modulus N occurring in $\tilde{\mathcal{F}}$ and $a > 1$.

Proof: We regard the polynomials $f \in \mathcal{F}$, the set induced by $\tilde{\mathcal{F}}$, as polynomials in $\mathbb{Z}[x_1, \ldots, x_l, t_1, \ldots, t_{|\mathcal{F}|}]$. From the precondition using Theorem 5.1.5 it follows that there are $\bar{f} \in \mathcal{F}$, $c_f \in \mathbb{Z}$ for all $f \in \mathcal{F} \setminus \{\bar{f}\}$ and $1 \neq a \in \mathbb{N}$ such that $\sum_{f \in \mathcal{F} \setminus \{\bar{f}\}} c_f f \equiv \bar{f} \pmod{a}$. By construction there is a monomial t_j which occurs only in $\bar{f}$. Its coefficient corresponds to the modulus N of the respective polynomial equation. Then for the linear dependency relation to hold we require $a|N$ which implies the claim. ■

In many practical examples we take moduli N which are products of two unknown primes p and q. Thus, we can only test for modular dependence modulo N. Like in Section 5.2 we, therefore, assume that we do not get the factors of the moduli as elementary divisors. Otherwise, this would be a way to determine the factorization of N. This assumption was true in all our examples.

Note that the analysis of systems of equations with mutually coprime moduli is thus easier than the analysis of systems of equations with the same modulus. From Lemma 5.3.2 we have that modular dependence only has to be checked with respect to the different moduli N_i and their divisors. A set of polynomials $\mathcal{P} \subseteq \mathcal{F}$ modularly dependent modulo a divisor a of N_i will only include shift polynomials which are multiples of f_i. This is because any other shift polynomial $f \in \mathcal{F}$ will include a term Nt such that t only occurs in this polynomial (as it is constructed by transforming $\tilde{f}$ to f) and $\gcd(N, a) = 1$. Consequently, the coefficient of t would not be $0 \pmod{a}$. This observation simplifies the analysis as we can take arbitrary shifts of $f_i^{\lambda_i}(x_1, \ldots, x_l)$, $i = 1, \ldots, k$, $\lambda_i \in \mathbb{N}$, without taking care of shared monomials. Only if we consider products of the polynomials as well, we have to be more careful.
We start the analysis of systems with mutually coprime moduli by returning to the case of univariate equations.

5.3.1 Directly Applying Coppersmith's Lattice-Based Techniques to a System of Equations to Solve SMUPE2

Let us recall the SMUPE2-problem which has been presented in Chapter 3: Let

$$\begin{array}{rcll} f_1(x) & \equiv & 0 & \pmod{N_1} \\ f_2(x) & \equiv & 0 & \pmod{N_2} \\ & \vdots & & \\ f_k(x) & \equiv & 0 & \pmod{N_k} \end{array}$$

be a system of univariate polynomial equations. Here, $N_1 < N_2 < \ldots < N_k$ are pairwise coprime composite numbers of unknown factorization and $f_1(x), \ldots, f_k(x)$ are polynomials of degree $\delta_1, \ldots, \delta_k$ in $\mathbb{Z}_{N_1}[x], \ldots, \mathbb{Z}_{N_k}[x]$, respectively.

Let $X < N_1$, $X \in \mathbb{R}$. The aim is to find all common roots x_0 of these equations with size $|x_0| \leq X$. In order to be able to calculate all roots, we would like to have $X = \frac{1}{2}N_1$.
This problem is a special case of the general problem of solving systems of equations with mutually coprime moduli as we are restricted to one variable.
In Chapter 3, Corollary 3.2.3, we have shown that if $\sum_{i=1}^{k} \frac{1}{\delta_i} \geq 1$, then we can determine all $x_0 \in \mathbb{Z}_{N_1}$ which are solutions of the system. To obtain this result, we have combined the polynomials $f_1, \ldots, f_k$ to one polynomial f with the same solutions modulo some product N of powers of the original moduli. The equation $f(x) \equiv 0 \pmod{N}$ could then be solved provably by a direct application of Coppersmith's algorithm.
Now we will analyze the problem differently. We will directly apply Coppersmith's algorithm to the system of equations. This implies that we first have to define a suitable set of shift polynomials. This choice also gives a general insight into applying Coppersmith's algorithm to systems of modular equations with pairwise coprime moduli.

We will give a different proof of the bound given in Corollary 3.2.3.

Proof of Corollary 3.2.3: In a first step, we define a set of shift polynomials and corresponding moduli $\tilde{\mathcal{F}}$ to use with Coppersmith's algorithm. Then we show that it is determinant preserving and prove that we get a new equation with zero x_0, which is valid over the integers, if $|x_0| \leq X = \frac{1}{2}N_1^{1-\epsilon}$ for an $\epsilon > 0$. Then we will extend the result to $x_0 \in \mathbb{Z}_{N_1}$.
Let $\epsilon > 0$, $\delta := \operatorname{lcm}(\delta_1, \ldots, \delta_k)$ and $\lambda \in \mathbb{N}$ such that $\lambda \geq \max\left\{\frac{k-1+\epsilon}{\epsilon\delta}, \frac{7}{\delta}\right\}$.
Recall that in order to achieve a large bound on the unknown we essentially need many powers of N_i and few powers of X^{-1} in the determinant of our lattice. As the lattice basis is constructed such that it is described by an upper triangular basis matrix, we require many powers of N_i and few powers of X^{-1} on the diagonal of the lattice basis.
Recall further that a monomial X^{-l} occurs on the diagonal if x^l is a monomial in any polynomial in $\mathcal{F}$. To obtain a good ratio of powers of X^{-1} and N_i, we simply reuse the parameter λ given by Coppersmith. This leads to the bound given above. We limit the degree of X^{-i} to $\lambda\delta$. Additionally, we would like to have as many powers of N_i as possible

contributing to the determinant. For any polynomial $f_i(x)$ we have $f_i(x) \equiv 0 \pmod{N_i}$. Thus, $f_i^j(x) \equiv 0 \pmod{N_i^j}$ for any $j \in \mathbb{N}$. Therefore, whenever introducing a new monomial we take the largest power of $f_i(x)$ with which it is possible to do so. We define

$$\tilde{\mathcal{F}} := \{(x^h f_i(x)^j, N_i^j) \mid i = 1, \dots, k; \quad 0 \le h < \delta_i \text{ and } 1 \le j < \lambda \frac{\delta}{\delta_i}\}\,.$$

This set is determinant preserving. To prove this, let $\mathcal{F}$ denote the set of integer polynomials induced by $\tilde{\mathcal{F}}$ and assume the contrary. Then, by Lemma 5.3.2 there are $\bar{f} \in \mathcal{F}$ and coefficients $c_f \in \mathbb{Z}$ not all equal to zero and $1 \neq a \in \mathbb{N}$ such that $\sum_{f \in \mathcal{F} \setminus \{\bar{f}\}} c_f f \equiv \bar{f}$ (mod a). Furthermore, a divides one of the moduli, i. e. $a | N_i$ for some index i. Due to our construction the moduli are either powers of the same value N_i or coprime to it. Consequently, linear dependence may only occur in a set of polynomials constructed with the same index i. Without loss of generality we assume this index to be 1. For any other index, we can argue analogously. Let $\tilde{\mathcal{F}}_1 := \{(x^h f_1(x)^j, N_1^j) \mid 0 \le h < \delta_1 \text{ and } 1 \le j < \lambda \frac{\delta}{\delta_1}\}$ and let $\mathcal{F}_1$ be induced by this set. Thus, $\sum_{f \in \mathcal{F}_1 \setminus \{\bar{f}\}} c_f f - \bar{f} \equiv 0 \pmod{a}$ with $a | N_1$. Ordering the polynomials in $\mathcal{F}_1$ by their degree in x, however, one can see that each polynomial introduces a new monomial. Thus, it cannot be dependent on the other polynomials. This contradicts the assumption. Hence, $\tilde{\mathcal{F}}$ is determinant preserving.

To determine a solution, we apply Coppersmith's method with shift polynomial set $\tilde{\mathcal{F}}$. We denote the lattice constructed with respect to $\tilde{\mathcal{F}}$ by L. In order to calculate the upper bound X up to which size we can determine the unknown x_0, we calculate the determinant of our lattice L. Its basic structure is $\det(L) = X^{-s_x} \prod_{i=1}^k N_i^{s_i}$. This is because the part induced by the shift polynomials set contributes with powers of the moduli, whereas from the part corresponding to the monomials we get powers of X^{-1}. It is $\mathbf{D} = \text{Diag}(1, X^{-1}, \dots, X^{-(\delta\lambda - 1)})$. Thus,

$$s_x = \sum_{i=0}^{\delta\lambda - 1} i = \frac{(\delta\lambda)(\delta\lambda - 1)}{2}\,.$$

For $1 \le i \le k$ we have

$$s_i = \delta_i \sum_{i=1}^{\lambda \frac{\delta}{\delta_i} - 1} i = \frac{\lambda\delta \left(\lambda \frac{\delta}{\delta_i} - 1\right)}{2}\,.$$

Consequently,

$$\det(L) = X^{-\left(\frac{(\delta\lambda)(\delta\lambda-1)}{2}\right)} \prod_{i=1}^k N_i^{\left(\frac{\lambda\delta\left(\lambda \frac{\delta}{\delta_i} - 1\right)}{2}\right)}\,.$$

Note that $\sqrt{\delta\lambda}$ denotes an upper bound on the size of the target vector. Using equation (2.13), we, therefore, get the following condition:

$$\sqrt{\delta\lambda} \le \left(X^{-\left(\frac{(\delta\lambda)(\delta\lambda-1)}{2}\right)} \prod_{i=1}^k N_i^{\left(\frac{\lambda\delta\left(\lambda \frac{\delta}{\delta_i} - 1\right)}{2}\right)} \right)^{\frac{1}{\delta\lambda}} 2^{-\frac{\delta\lambda - 1}{4}}\,.$$

This is equivalent to the condition

$$X \leq 2^{-\frac{1}{2}} (\delta\lambda)^{-\frac{1}{\delta\lambda-1}} \prod_{i=1}^{k} N_i^{\frac{\lambda\frac{\delta}{\delta_i}-1}{\delta\lambda-1}} .$$

As $\delta\lambda \geq 7$ we can conclude $(\delta\lambda)^{-\frac{1}{\delta\lambda-1}} \geq 7^{-\frac{1}{6}} \geq 8^{-\frac{1}{6}} = 2^{-\frac{1}{2}}$.
Further, as $\lambda \geq \frac{k-1+\epsilon}{\epsilon\delta} \Leftrightarrow \epsilon \geq \frac{k-1}{\delta\lambda-1}$ and $\sum_{i=1}^{k} \frac{1}{\delta_i} \geq 1$, it holds that

$$\begin{aligned} \sum_{i=1}^{k} \frac{\lambda\frac{\delta}{\delta_i}-1}{\delta\lambda-1} &= \sum_{i=1}^{k} \left(\frac{1}{\delta_i} - \frac{1-\frac{1}{\delta_i}}{\delta\lambda-1} \right) \\ &= \sum_{i=1}^{k} \frac{1}{\delta_i} - \frac{1}{\delta\lambda-1} \sum_{i=1}^{k} \left(1 - \frac{1}{\delta_i}\right) \\ &= \sum_{i=1}^{k} \frac{1}{\delta_i} - \frac{k - \sum_{i=1}^{k} \frac{1}{\delta_i}}{\delta\lambda-1} \\ &\geq \sum_{i=1}^{k} \frac{1}{\delta_i} - \frac{k-1}{\delta\lambda-1} \\ &\geq \sum_{i=1}^{k} \frac{1}{\delta_i} - \epsilon . \end{aligned}$$

Consequently,

$$\begin{aligned} 2^{-\frac{1}{2}} (\delta\lambda)^{-\frac{1}{\delta\lambda-1}} \prod_{i=1}^{k} N_i^{\frac{\lambda\frac{\delta}{\delta_i}-1}{\delta\lambda-1}} &\geq \frac{1}{2} N_1^{\sum_{i=1}^{k}\frac{1}{\delta_i}-\epsilon} \\ &\geq \frac{1}{2} N_1^{1-\epsilon} . \end{aligned}$$

For $X = \frac{1}{2}N_1^{1-\epsilon}$ we can, thus, determine $f(x)$ with Coppersmith's algorithm such that $f(x) = 0$ over the integers. This requires time polynomial in $\delta, \max\{\log(N_i)\}$ and $\frac{1}{\epsilon}$.
In order to determine all solutions $x_0 \in \mathbb{Z}_{N_1}$, we have to redo the algorithm. Setting e. g. $\epsilon = \frac{1}{\log N_1}$ we can recover all x_0 in the interval $[-\frac{1}{4}N_1, \frac{1}{4}N_1]$. To cover a complete set of representatives $\pmod{N_1}$, we additionally need to cover e. g. the interval $[\frac{1}{4}N_1, \frac{3}{4}N_1]$. Therefore, we repeat the above algorithm with a set of polynomial equations $\tilde{f}_i(x+\bar{x}) \equiv 0 \pmod{N_i}$, $i = 1, \ldots, k$ centered at $\bar{x} := \lfloor \frac{1}{2}N_1 \rfloor$. The computation of the new polynomials is polynomial in $\delta_i \leq \delta$. Thus, we can determine any solution x_0 of our given system in time polynomial in k, δ and $\max\{\log(N_i)\}$. ■

The construction described here shows an easy way to construct a shift polynomial set based on a set of univariate modular equations. This set seems to be a good choice as we recalculate the same bounds we have obtained in the proof of Corollary 3.2.3 by using

the Chinese Remainder Theorem 2.2.13. Moreover, we have argued in Section 3.2.1 that this bound is optimal for general problems. With the Chinese Remainder Theorem the optimality is quite intuitive. First, the polynomials are taken to powers. The bound on the zero we can obtain when analyzing a single polynomial is not influenced by this. Then the Chinese Remainder Theorem is applied. It gives a good construction to obtain one polynomial of common degree. Thus, the degree remains, as modulus we get the product of the single moduli. Coppersmith's algorithm, which is then applied to this polynomial, is believed to be optimal as we have explained in the introduction of Section 3.
Using the method presented above, however, the optimality is not obvious as the choice of a "good" set of shifts for systems of equations is not that clear. Let us consider possible changes of the shift polynomial set $\tilde{\mathcal{F}}$. As the value of λ is optimized afterwards, we do not need to think about including shifts of higher powers of $f_i(x)$. Another idea to increase the determinant is to add further shift polynomials, e. g. products of polynomials $f_i(x)f_l(x)$ of small enough degree. They do not introduce new monomials on the diagonal but contribute with powers of the N_i. Adding such products to $\tilde{\mathcal{F}}$ and thereby to $\mathcal{F}$, however, smashes the property of being determinant preserving of $\mathcal{F}$ (and thus $\tilde{\mathcal{F}}$) as can be seen in the following lemma.

Lemma 5.3.3
Let $\tilde{\mathcal{F}} := \{(x^h f_i(x)^j, N_i^j) \mid i = 1,\dots,k;\quad 0 \le h < \delta_i \text{ and } 1 \le j < \lambda\frac{\delta}{\delta_i}\}$ and $\tilde{f} := x^h \prod_{i=1}^k f_i^{\lambda_i}(x)$ with $\lambda_i \in \mathbb{N}$ for at least two indices i and $\lambda_i = 0$ for all other indices. Furthermore, let $\deg \tilde{f} < \delta\lambda$. Then the set $\tilde{\mathcal{F}}_n := \tilde{\mathcal{F}} \cup \{(\tilde{f}, \prod_{i=1}^k N_i^{\lambda_i})\}$ is not determinant preserving.

Before proving this, we give an auxiliary lemma.

Lemma 5.3.4
Let $\lambda_1, N_1 \in \mathbb{N}$. Let $f(x), f_1(x) \in \mathbb{Z}_{N_1}[x]$ be polynomials such that

$$f(x) \equiv f_1^{\lambda_1}(x)\cdot g(x) \pmod{N_1},$$

where $g(x) \in \mathbb{Z}_{N_1}[x]$ is a polynomial of degree $\delta_g := \deg(g) = \deg(f) - \lambda_1\deg(f_1) \ge 1$. Let $\delta_f := \deg(f)$ and $\deg(f_1) =: \delta_1$. Furthermore, let c_f be the leading coefficient of $f(x)$.
Then $h(x) := f(x) - c_f x^{\delta_f - \lfloor\frac{\delta_f}{\delta_1}\rfloor\delta_1} f_1^{\lfloor\frac{\delta_f}{\delta_1}\rfloor}(x) \in \mathbb{Z}_{N_1}[x]$ is a polynomial of degree $\delta_h := \deg(h) < \delta_f$. Moreover, either $h(x) \equiv f_1^{\lambda_1}(x)\cdot g_h(x) \pmod{N_1}$ for a non-zero polynomial $g_h(x)$ or $h(x) \equiv 0 \pmod{N_1}$.

Proof: The proof of this lemma is straightforward. It is $f(x) \equiv c_f x^{\delta_f} + f_r(x) \pmod{N_1}$ with $\deg(f_r) < \delta_f$ as $c_f x^{\delta_f}$ denotes the leading term of f. Similarly, $x^{\delta_f - \lfloor\frac{\delta_f}{\delta_1}\rfloor\delta_1} f_1^{\lfloor\frac{\delta_f}{\delta_1}\rfloor}(x) \equiv x^{\delta_f} + f_{r1}(x) \pmod{N_1}$ with $\deg(f_{r1}) < \delta_f$. Thus, $h(x) := f(x) - c_f x^{\delta_f - \lfloor\frac{\delta_f}{\delta_1}\rfloor\delta_1} f_1^{\lfloor\frac{\delta_f}{\delta_1}\rfloor}(x) \equiv f_r(x) - c_f f_{r1}(x) \pmod{N_1}$. By this we have $\delta_h < \delta_f$.
By construction, we have $f_1^{\lambda_1}(x)|f(x)$. Further, it is $\delta_f > \lambda_1\delta_1 \Leftrightarrow \frac{\delta_f}{\delta_1} > \lambda_1$. As λ_1

is an integer, even $\lfloor \frac{\delta_f}{\delta_1} \rfloor \geq \lambda_1$ holds. Therefore, $f_1^{\lambda_1}(x) | x^{\delta_f - \lfloor \frac{\delta_f}{\delta_1} \rfloor \delta_1} f_1^{\lfloor \frac{\delta_f}{\delta_1} \rfloor}(x)$. Consequently, $f_1^{\lambda_1}(x) | h(x)$, i.e. $h(x) \equiv f_1^{\lambda_1}(x) \cdot g_h(x) \pmod{N_1}$ for a non-zero polynomial $g_h(x)$ or $h(x) \equiv 0 \pmod{N_1}$. ■

Now we can give the proof of Lemma 5.3.3:
PROOF: Let $\mathcal{F}_n$ be the integer polynomial set induced by $\tilde{\mathcal{F}}_n$. To show that $\tilde{\mathcal{F}}_n$ is not determinant preserving, we will show that $\mathcal{F}_n$ is not determinant preserving. Let $f := \tilde{f} - t \prod_{i=1}^{k} N_i^{\lambda_i}$ be the integer polynomial induced by $\tilde{f}$. Without loss of generality we assume $\lambda_1 > 0$. (If not, enumerate the (f_i, N_i) differently. This can be done as this proof does not depend on the fact that the N_i are ordered according to their size.) We will show that f is linearly dependent on $\mathcal{F}$ modulo N_1. Then, by Theorem 5.1.5, $\mathcal{F}_n$ is not determinant preserving.
Let $\delta_1 := \deg(f_1)$ and $\delta_f := \deg(f)$. By construction we have $f(x) \equiv \tilde{f}(x) \equiv f_1^{\lambda_1}(x) \cdot g(x) \pmod{N_1}$ with $\delta_g := \deg(g) = \delta_f - \lambda_1 \delta_1 > 0$.
By iteratively applying Lemma 5.3.4 on f, we get a sequence $f = h_1, h_2, \ldots, h_\nu \in \mathbb{Z}_{N_1}[x]$ of polynomials of decreasing degree. It is finite as the degree decreases in each step. Let ν be defined such that $h_\nu \equiv 0$, but $h_{\nu-1} \not\equiv 0$. Further, for $i = 1, \ldots, \nu - 1$, it is $h_i(x) \equiv h_{i+1}(x) + c_{h_i} x^{\deg(h_i) - \lfloor \frac{\deg(h_i)}{\delta_1} \rfloor \delta_1} f_1^{\lfloor \frac{\deg(h_i)}{\delta_1} \rfloor}(x) \pmod{N_1}$. Successively substituting the $h_i(x)$, we get

$$f(x) \equiv \sum_{i=1}^{\nu-1} c_{h_i} x^{\deg(h_i) - \lfloor \frac{\deg(h_i)}{\delta_1} \rfloor \delta_1} f_1^{\lfloor \frac{\deg(h_i)}{\delta_1} \rfloor}(x) \pmod{N_1}.$$

This shows that $\mathcal{F}_n$ is not determinant preserving. ■

Remark that in the above proof the representation of f as a linear combination of polynomials in $\mathcal{F}$ only includes polynomials of the form $x^h f_1^j(x)$ with $j \geq \lambda_1$. Thus, with the same proof we get a more specific version of Lemma 5.3.3.

Lemma 5.3.5
Let $\tilde{\mathcal{F}}_{1,\lambda_1} := \{(x^h f_1(x)^j, N_1^j) \mid 0 \leq h < \delta_i$ *and* $\lambda_1 \leq j < \lambda \frac{\delta}{\delta_i}\}$ *and* $\tilde{f} := x^h \prod_{i=1}^{k} f_i^{\lambda_i}(x)$ *with* $\lambda_1, \lambda_\iota \in \mathbb{N}$ *with* $\iota \neq 1$ *and* $\lambda_i \in \mathbb{N}_0$ *for all other indices. Furthermore, let* $\deg \tilde{f} < \delta\lambda$. *Then the set* $\tilde{\mathcal{F}}_n := \tilde{\mathcal{F}}_{1,\lambda_1} \cup \{(\tilde{f}, \prod_{i=1}^{k} N_i^{\lambda_i})\}$ *is not determinant preserving.*

We have seen that we cannot include any further polynomial in the shift polynomial set

$$\tilde{\mathcal{F}} = \{(x^h f_i(x)^j, N_i^j) \mid i = 1, \ldots, k; \quad 0 \leq h < \delta_i \text{ and } 1 \leq j < \lambda \frac{\delta}{\delta_i}\}$$

in order to improve the bound. Nevertheless, a completely different choice of $\tilde{\mathcal{F}}$, not including all the polynomials we have used so far, might lead to an improved bound. Therefore, we have to check which bounds can be obtained if we use different shift polynomial sets. Let us assume that any shift polynomial set under consideration contains all monomials x^i with $i \leq \delta$, where δ is the maximum degree of a polynomial in $\tilde{\mathcal{F}}$. This is a sensible

assumption as the bound obtained with Coppersmith's method for univariate polynomials only depends on the degree of the original polynomial and not on its sparseness. The lack of monomials does not influence the asymptotic behavior.
A direct analysis shows that any arbitrary determinant preserving shift polynomial set can be substituted by a shift polynomial set which is a subset of our original set $\tilde{\mathcal{F}}$.

Lemma 5.3.6
Let $\lambda, \delta \in \mathbb{N}$. Let

$$\tilde{\mathcal{G}}_1 \subseteq \{(x^h \prod_{i=1}^{k} f_i^{\lambda_i}(x), \prod_{i=1}^{k} N_i^{\lambda_i}) \mid i = 1, \ldots, k; h, \lambda_i \in \mathbb{N}_0 \text{ such that } \sum_{i=1}^{k} \lambda_i \delta_i + h < \delta\lambda\}$$

be a determinant preserving shift polynomial set. Let $Mon(\tilde{\mathcal{G}}_1)$ include all monomials up to the maximum degree of a polynomial in $\tilde{\mathcal{F}}$. Then there exists another determinant preserving shift polynomial set

$$\tilde{\mathcal{G}}_2 \subseteq \tilde{\mathcal{F}} := \{(x^h f_i(x)^j, N_i^j) \mid i = 1, \ldots, k; \quad 0 \leq h < \delta_i \text{ and } 1 \leq j < \lambda \frac{\delta}{\delta_i}\}$$

with which we obtain at least the same upper bound X on the size of the solution we can determine.

Proof: Let $\mathcal{G}_1$ be the integer polynomial set induced by $\tilde{\mathcal{G}}_1$.Let $(\tilde{g}, N) \in \tilde{\mathcal{G}}_1$ such that $\tilde{g}$ is the polynomial of smallest degree which is a product of at least two different polynomials $f_i(x)$, $1 \leq i \leq k$. Then $\tilde{g}$ can be written as $\tilde{g}(x) = x^h \prod_{i=1}^{k} f_i^{\lambda_i}(x)$ and $N_g := \prod_{i=1}^{k} N_i^{\lambda_i}$ with $\lambda_i \in \mathbb{N}$ for at least two indices i and $\lambda_i = 0$ for all other indices. Furthermore, $\deg(\tilde{g}) < \delta\lambda$. Without loss of generality we assume $\lambda_1 > 0$. (If not, enumerate the f_i differently.) Let $g(x)$ be the integer polynomial induced by $\tilde{g}(x)$, that is $g(x, t) := \tilde{g}(x) - tN_g$.
We would like to substitute $\tilde{g}$ by a polynomial or a set of polynomials belonging to $\tilde{\mathcal{F}}$. In order to do so, we proceed in two major steps: First, we show that there exist polynomials which we can take for the substitution, i. e. polynomials which are not yet in $\mathcal{G}_1$. Then we show that these polynomials either keep or improve the bound, but do not worsen it. The proof of this consists of two steps. On the one hand, we show that the positive contribution, i. e. the contribution of powers of N_i to the determinant, is equal to the contribution when using g. On the other hand, we need to have that the negative contribution, i. e. the contribution of powers of X^{-1} to the determinant, is smaller than or equal to the contribution when using g. This is already given by the assumption that $\text{Mon}(\tilde{\mathcal{G}}_1)$ contains all monomials. Hence, no further monomials can be introduced by substituting the polynomial. Thus, the value of the determinant can only increase and, consequently, the same holds for the bound.
We now prove the first claim. Let $\tilde{\mathcal{F}}_{1,\lambda_1}$ denote the set of all polynomials in $\mathcal{F}$ which are divisible by $f_1^{\lambda_1}(x)$ as in Lemma 5.3.5. By Lemma 5.3.5 we have that $\tilde{\mathcal{F}}_{1,\lambda_1} \cup \{\tilde{g}\}$ is not determinant preserving. Consequently, as $(\tilde{g}, N_g) \in \tilde{\mathcal{G}}_1$ there exists $(\tilde{g}_1, N_{g_1}) \in \tilde{\mathcal{F}}_{1,\lambda_1} \setminus \tilde{\mathcal{G}}_1$ with $\tilde{g}_1(x) \equiv f_1^{\lambda_1}(x) g_{r1}(x) \pmod{N_1^{\lambda_1}}$. Analogously, for all other i such that $\lambda_i > 0$ there

exist polynomials $\tilde{g}_i(x) \equiv f_i^{\lambda_i}(x) g_{ri}(x) \pmod{N_i^{\lambda_i}}$ which are not contained in $\tilde{\mathcal{G}}_1$. We then substitute $(\tilde{g}, N_g)$ by the set $\{(\tilde{g}_i, N_i^{\lambda_i}) \mid \lambda_i > 0\}$.
Now we consider the influence of this substitution on the determinant of the lattice. The contribution of N_i to the determinant using $\tilde{g}$ is $N_g := \prod_{i=1}^k N_i^{\lambda_i}$. That is, for each $\lambda_i > 0$, we get a factor of $N_i^{\lambda_i}$ in the determinant. For each of the $\tilde{g}_i$, the contribution to the determinant is $N_i^{\lambda_i}$. The product of these contributions is, thus, $\prod_{i=1}^k N_i^{\lambda_i}$. Hence, the substitution does not change the power of N_i occurring in the determinant.
Moreover, as $\tilde{\mathcal{F}}$ is determinant preserving, $\tilde{\mathcal{G}}_2 \subseteq \tilde{\mathcal{F}}$ is determinant preserving as well. ■

We have seen that a shift polynomial set which is a subset of $\tilde{\mathcal{F}}$ can be used to analyze systems of modular univariate equations with coprime moduli. As we have assumed the sets to contain all monomials up to a certain degree anyway, it is best to take as many polynomials as possible. This implies that the bound using $\tilde{\mathcal{F}}$ is better than the bound taking a subset. Therefore, on the assumption that any useful shift polynomial set includes all monomials up to a certain degree, we conclude in the univariate case:

Theorem 5.3.7
The following two analyses of the SMUPE2 problem give the same bounds:

1. *Combining the equations by the Chinese Remainder Theorem to a polynomial $f(x)$ and applying Coppersmith's algorithm to $f(x)$.*

2. *Defining a good shift polynomial set like $\tilde{\mathcal{F}}$ for the system of equations and applying Coppersmith's algorithm to this.*

Both analyses can be performed in time polynomial in δ, k and $\max\{\log(N_i)\}$.

PROOF: By the proof of Corollary 3.2.3 given in this section we get that any bound obtained with method (1) can also be obtained directly using a suitable shift polynomial set. For the converse, assume we are given a shift polynomial set $\tilde{\mathcal{F}}_1$ which gives a good bound. By Lemma 5.3.6 we get a second shift polynomial set $\tilde{\mathcal{F}}_2 \subset \tilde{\mathcal{F}}$ which gives the same (or even a better) bound. As the polynomials in $\tilde{\mathcal{F}}$ contain all monomials, the shift polynomial set $\tilde{\mathcal{F}}$ gives a better bound than any of its subsets. Taking these shifts, however, is equivalent to combining the original polynomials by the Chinese Remainder Theorem and shifting the resulting polynomial f up to a power f^λ.
The running time of method (1) is polynomial in δ and $\log M$ by Theorem 3.2.2. The value M is defined as $M := \prod_{i=1}^k N_i^{\frac{\delta}{\delta_i}}$. Hence, the running time is polynomial in δ, k and $\max\{\log(N_i)\}$. Regarding the second method, we directly get the running time by the proof of Corollary 3.2.3 given in this section. This concludes the proof. ■

We are interested in generalizing the observations made during the analysis of the SMUPE2-problem to systems of multivariate equations. The analysis gets more complicated as we can no longer assume the shift polynomial set to contain all monomials. It would be a vast overload to assume that a polynomial contains all monomials up to a certain degree. There may even be variables which do not occur in all of the equations. Nevertheless, as

a first approach we return to the method used in Section 3.2. We combine the given set of equations by the Chinese Remainder Theorem and obtain a polynomial f. Then we pursue the standard analysis of a single multivariate modular equation $f(x_1, \ldots, x_l) \equiv 0 \pmod{N}$ given in [JM06]. That is, we take the monomials of the Newton polytope to shift the powers of f.
The same result can again be obtained by an analysis using a corresponding shift polynomial set. The proof in this case is analogous to the proof of Corollary 3.2.3 in Section 5.3.1. Unfortunately, an analogue to Lemma 5.3.6 would no longer be interesting as the underlying assumption is too strong in the multivariate case.
The following example shows that the bounds we obtain by analyzing the polynomial f instead of the system of equations are far from optimal.

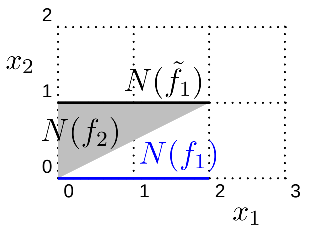

Figure 5.1: Newton polytopes of f_1, f_2 and $\tilde{f}_1$ of Example 5.3.8.

Example 5.3.8
Let

$$\begin{aligned} f_1(x_1, x_2) := x_1^2 + x_1 + 1 &\equiv 0 \pmod{N_1} \\ f_2(x_1, x_2) := x_1^2 x_2 + x_2 + 1 &\equiv 0 \pmod{N_2} \end{aligned}$$

be a system of equations with the solution $(\bar{x}_1, \bar{x}_2)$. As $f_1(x_1, x_2)$ is a univariate equation in $\mathbb{Z}_{N_1}[x_1]$, we know from Theorem 2.3.9 that we can determine all solutions $\bar{x}_1$ such that $|\bar{x}_1| < N_1^{\frac{1}{2}}$. Substituting the value x_1 by $\bar{x}_1$ in f_2, we get the equation $f_2(\bar{x}_1, x_2) = (\bar{x}_1^2 + 1)x_2 + 1 \equiv 0 \pmod{N_2}$, which is linear in x_2. Thus, we can determine all solutions $\bar{x}_2 \in \mathbb{Z}_{N_2}$ such that $f_2(\bar{x}_1, \bar{x}_2) \equiv 0 \pmod{N_2}$.
Consequently, in a combined analysis via Coppersmith's method we would expect to obtain the condition $X_1 X_2 < N_1^{\frac{1}{2}} N_2$, where X_i denotes an upper bound on the size of $|\bar{x}_i|$, $i = 1, 2$. We apply the Chinese Remainder Theorem to combine f_1 and f_2. Let $f(x_1, x_2) = (N_2^{-1} \pmod{N_1}) N_2 f_1(x_1, x_2) + (N_1^{-1} \pmod{N_2}) N_1 f_2(x_1, x_2) \equiv 0 \pmod{N_1 N_2}$ denote the resulting polynomial equation. The bound we obtain if we analyze the polynomial f with standard shift polynomial sets using Coppersmith's method, however, is worse than the trivial bound. The Newton polytope of the polynomial f is a rectangle. Thus, applying the basic strategy of [JM06] for generalized rectangles we obtain the bound

$$X_1^3 X_2^{\frac{3}{2}} < N_1 N_2 \,.$$

Hence, directly applying the Chinese Remainder Theorem to a system of multivariate equations will not generally lead to good bounds. Having a look at the Newton polytopes of f_1 and f_2, this is not really surprising. The Newton polytope of f_1 forms a line on the x-axis whereas the Newton polytope of f_2 is a triangle which leaves out the x-axis. The only monomial which is shared by both polynomials is, thus, the constant term. Consequently, by combining the two polynomials via Chinese Remaindering, we get several superfluous monomials. To overcome these difficulties, we aim at constructing polynomials which share many monomials. Let us consider the polynomial $\tilde{f}_1(x_1, x_2) := x_2 f_1(x_1, x_2) = x_1^2 x_2 + x_1 x_2 - x_2$. Its Newton polytope is contained in the Newton polytope of f_2. By the Chinese Remainder Theorem we combine $\tilde{f}_1$ and f_2 and get $g(x_1, x_2) \equiv (N_2^{-1} \pmod{N_1}) N_2 \tilde{f}_1(x_1, x_2) + (N_1^{-1} \pmod{N_2}) N_1 f_2(x_1, x_2)$. To determine the roots of $g \pmod{N_1 N_2}$ we can again use Coppersmith's method with the basic strategy of [JM06]. The Newton polytope is a generalized triangle. Therefore, we obtain the bound

$$X_1^2 X_2 < N_1 N_2 \,. \tag{5.20}$$

This bound corresponds to the one we have expected beforehand. More precisely, $X_1 < N_1^{\frac{1}{2}}$ and $X_2 < N_2$ imply $X_1^2 X_2 < N_1 N_2$. The opposite implication, however, does not hold. This is quite natural as f_2 comprises both unknowns. A separate analysis using this polynomial would, thus, allow to solve for any value $|\bar{x}_1| \leq X_1$ as long as $X_1^2 X_2 < N_2$. This is the result we also obtain by the analysis of $f_2(x_1, x_2) \equiv 0 \pmod{N_2}$.

A general approach to solve systems of multivariate equations with coprime moduli, consequently, has to be adapted to the specific system of equations. The following basic strategy will be quite helpful. First, transform the original polynomials (by powering or multiplications) such that the new polynomials have a shared Newton polytope. Then combine the equations by the Chinese Remainder Theorem to a polynomial g and determine its solutions with standard Coppersmith techniques.

5.4 General Systems of Modular Equations

In the previous sections of this chapter systems of modular equations either all sharing the same modulus or all having mutually coprime moduli were analyzed. These techniques can be combined to solve any system of modular equations. For arbitrary $N_1 \leq \ldots \leq N_k \in \mathbb{N}$ we are given the following system of equations:

$$\begin{array}{rcl} f_1(x_1, \ldots, x_l) & \equiv & 0 \pmod{N_1} \\ f_2(x_1, \ldots, x_l) & \equiv & 0 \pmod{N_2} \\ & \vdots & \\ f_k(x_1, \ldots, x_l) & \equiv & 0 \pmod{N_k} \,. \end{array} \tag{5.21}$$

Without loss of generality we assume that all moduli are indeed equal or coprime. If not, we can compute their greatest common divisor. Then we can apply the Chinese Remainder

Theorem and substitute equations by a system of isomorphic ones. This can be performed iteratively until we have a system of equations which fulfills the requirement.
Then we can arrange the equations by common or mutually coprime moduli. For any group of equations which share the same modulus, we test by Groebner basis computation if the resulting system contains a linear univariate polynomial. If yes, we determine the corresponding solutions $\bar{x}_i$ of x_i and substitute them in all equations. By this, we get a system of equations in less variables.
As a next step, we choose possibly helpful sets of equations with mutually coprime moduli and apply the techniques from the previous section.
An alternative way to analyze the given system is to directly define a combined shift polynomial set $\tilde{\mathcal{F}}$ and then apply Coppersmith's method with this shift polynomial set. Note however, that the choice of shift polynomials has to be made extremely carefully. Whereas any shifts of polynomials with coprime moduli can be taken and the set remains determinant preserving, shifts of polynomials with common moduli have to be taken very carefully. A possible choice is to let them introduce different monomials.
The bounds we obtain this way strongly depend on the specific system of equations and the choice of the special subsystems we choose to analyze. Therefore, we cannot state any general bounds here.

Chapter 6

Solving Systems of Multivariate Polynomial Equations over the Integers

In the previous chapter we have seen how to adapt Coppersmith's method to analyze systems of modular multivariate equations. The equations to describe specific problems, however, can also be integer equations. One possible way to analyze them is to regard them as modular equations. An integer equation can be transformed into a modular one by taking one of its coefficients as modulus. However, information gets lost this way. In case of a single equation we may loose information on the relation of the unknowns. In case of a system of equations even more problems may occur. In the worst case, by the modular operation, we could eliminate the only variable which occurs in more than one equation. Therefore, we should adapt our analyses to systems of non-modular equations. To do so, we analyze *Systems of Integer Multivariate Polynomial Equations* (SIMPE-problem) in this chapter.

Definition 6.0.1 (SIMPE-problem)
Let $k \in \mathbb{N}$, $\delta_1, \ldots, \delta_k \in \mathbb{N}$. Assume $f_1(x_1, \ldots, x_l), \ldots, f_k(x_1, \ldots, x_l)$ to be polynomials of degree $\delta_1, \ldots, \delta_k$ in $\mathbb{Z}[x_1, \ldots, x_l]$, respectively. Let

$$\begin{array}{rcl} f_1(x_1, \ldots, x_l) & = & 0 \\ f_2(x_1, \ldots, x_l) & = & 0 \\ & \vdots & \\ f_k(x_1, \ldots, x_l) & = & 0 \end{array} \tag{6.1}$$

be a system of multivariate polynomial equations.

Let $X_i \in \mathbb{R}$, $i = 1, \ldots, l$. Find all common roots $(\bar{x}_1, \ldots, \bar{x}_l)$ of (6.1) with size $|\bar{x}_i| \leq X_i$.

All the systems of polynomial equations we have considered in the previous chapters can be interpreted as special cases of this system. Any modular equation $f(x_1, \ldots, x_l) \equiv 0 \pmod{N}$ can be written as an equation $f(x_1, \ldots, x_l) - t_f N = 0$ valid over the integers. Thus, any system of modular equations can be transformed into a system of equations

valid over the integers. The SIMPE-problem, however, includes further problems which are directly given as systems of integer polynomial equations. Our final goal is to determine bounds X_i such that we can compute all solutions smaller than these bounds.
We would like to calculate the solutions using Coppersmith's method. As the SIMPE-problem is the most general instance of a system of equations we would like to solve, its analysis is the most difficult one. Let $\mathcal{F}$ denote a shift polynomial set constructed using the polynomial equations given in (6.1), and let $\mathbf{F}$ be the matrix the columns of which correspond to the polynomials in $\mathcal{F}$. Recall that in the modular case the matrix $\mathbf{F}$ consists of an upper part $\mathbf{F_c}$ corresponding to the coefficients of the polynomials and a lower part $\mathbf{F_m}$ corresponding to the moduli. The matrix $\mathbf{F_m}$ is a diagonal matrix. Thus, $\mathbf{F}$ is an upper triangular matrix in the modular case.
However, as the polynomials we are considering now are valid over the integers, the matrix $\mathbf{F}$ can be of any shape. Given an arbitrary shift polynomial set, we do no longer have a triangular structure. To see this, let

$$\begin{array}{rcl} f_1(x_1,x_2) := -2x_1x_2 + 3x_1^3 + 4x_2^2 & = & 0 \\ f_2(x_1,x_2) := -3x_1x_2 + 2x_1^3 + x_2^2 & = & 0 \end{array} \tag{6.2}$$

be a system of equations with the solution $(\bar{x}_1, \bar{x}_2) = (-2, 2)$. Let $\mathcal{F} := \{f_1, f_2\}$ denote the shift polynomial set used to analyze the system of equations. The matrix $\mathbf{F}$ corresponding to this shift polynomial set is

$$\mathbf{F} := \begin{pmatrix} -2 & -3 \\ 3 & 2 \\ 4 & 1 \end{pmatrix}. \tag{6.3}$$

As $\mathbf{F}$ does not have a triangular structure, determinant calculations, and, consequently, calculations of bounds get complicated. Even if we manage to calculate the determinant of the lattice, we will have to check if it is also the determinant of the sublattice.
We have already analyzed the matrix $\mathbf{F}$ in Example 5.1.7. From that analysis we know that $\mathcal{F}$ is not determinant preserving as f_1 and f_2 are modularly dependent modulo 5. Recall that in order to check if an arbitrary shift polynomial set $\mathcal{F}$ is determinant preserving, we have to check if the polynomials in $\mathcal{F}$ are linear independent modulo N for any positive integer $1 < N < \left(\sqrt{|\mathcal{F}|} \cdot |c_{\max}|\right)^{|\mathcal{F}|}$, where $c_{\max}$ denotes the largest coefficient which occurs in $\mathcal{F}$. This is not efficiently possible.
Recall further that, in case of modular systems of equations, the number of potential moduli for linear dependence is restricted by the number of moduli (and their divisors) occurring in the system itself. These conditions were described in Lemma 5.2.2 with respect to common moduli and in Lemma 5.3.2 with respect to coprime moduli. For a general integer polynomial set, though, we only have the general condition of a determinant preserving set given in Theorem 5.1.5. This condition, however, is impossible to check for. Hence, we have to consider this problem beforehand. Namely, while constructing a shift polynomial set, we construct a shift polynomial set that allows for simpler checks if the set is determinant preserving and for easier calculations of the bounds. One possibility is

to construct the shift polynomial set in a way that a new monomial is introduced with each new polynomial. Then we obtain a simpler condition (Lemma 5.1.8). That is, while building a shift polynomial set with respect to a system of integer polynomial equations, we have to concentrate on two aspects: On the one hand, we always have to check if the set we define really is determinant preserving. On the other hand, we have to construct the shift polynomial set in a way that gives us a good bound.

Before we develop new strategies, let us present an existing method which comprises an analysis of a special system of equations over the integers. Aurélie Bauer and Antoine Joux have shown how to provably determine small solutions of a trivariate integer polynomial equation $p_1(x_1, x_2, x_3) = 0$ [BJ07, Bau08]. Their analysis consists of two basic steps. First, they determine a polynomial $p_2(x_1, x_2, x_3)$ by Coppersmith's basic method. By construction, the polynomial $p_2(x_1, x_2, x_3)$ is coprime to $p_1(x_1, x_2, x_3)$. In the second step of their analysis they determine a third polynomial $p_3(x_1, x_2, x_3)$ using p_1 and p_2 such that the system

$$p_i(x_1, x_2, x_3) = 0, \quad i = 1, 2, 3\,, \tag{6.4}$$

is zero dimensional. In order to determine p_3, they apply Coppersmith's method using a shift polynomial set $\mathcal{F}$ which they derive from p_1 and p_2. Let us briefly summarize the construction of p_3.
To prove the zero dimensionality of system (6.4), the polynomial p_3 is constructed such that $p_3 \notin I = \langle p_1, p_2 \rangle$. If p_1 and p_2 are coprime and I is prime, this condition is sufficient. To construct p_3, a minimal Groebner basis $G = \{q_1, \ldots, q_r\}$ of I is calculated. The polynomials $q_1, \ldots, q_r$ are then shifted by monomials from sets $S_1, \ldots, S_r$, respectively. Let $\mathcal{F} := \{m_i q_i \mid m_i \in S_i \text{ and } m_i LM(q_i) \neq m_j LM(q_j) \text{ for some } j < i\}$ denote the shift polynomial set determined this way, and let M denote the set of monomials which occur in $\mathcal{F}$. We would like to describe any polynomial $p \in I$ of bounded degree as a linear combination of polynomials of $\mathcal{F}$. We know $p = f_1 q_1 + \ldots f_r q_r$ with polynomials f_i. Thus, we require the set S_i to be a generating set of monomials to generate the polynomials f_i. This is captured by the notion of admissibility.

Definition 6.0.2
Let $(S_1, \ldots, S_r, M)$ denote non-empty sets of monomials in $\mathbb{Z}[x_1, x_2, x_3]$. Then the tuple $(S_1, \ldots, S_r, M)$ is called ***admissible*** *for an ideal I with the Groebner basis $\{q_1, \ldots, q_r\}$ if:*

1. *For all $(m_1, \ldots, m_r) \in S_1 \times \ldots \times S_r$ the polynomial $m_1 q_1 + \ldots + m_r q_r$ only contains monomials of M.*

2. *For any polynomial g containing only monomials of M which can be written as $g = f_1 q_1 + \ldots + f_r q_r$ it holds that f_i only consists of monomials of S_i for all i.*

Aurélie Bauer and Antoine Joux impose admissibility as a condition on $(S_1, \ldots, S_r, M)$ in their construction and build the sets accordingly. This, however, is not yet enough to prove the zero dimensionality of system (6.4). In order to be able to construct the sets S_i by polynomial division no new monomials may be introduced by the division process. This property is ensured by using a set M and an order $<$ which are compatible.

Definition 6.0.3
*The tuple $(M, <)$ is called **compatible** if $m_1 \in M$ implies $m_2 \in M$ for any monomial $m_2 < m_1$.*

On the aforementioned constraints, a polynomial p_3 constructed by Coppersmith's method is coprime to p_1 and p_2. Due to construction, p_1 and p_2 are coprime as well. Thus, system (6.4) is zero dimensional as required.
Furthermore, the preconditions given in [BJ07] already imply that the shift polynomial set $\mathcal{F}$ is determinant preserving. To see this, assume the contrary. Then there exists a linear combination of polynomials of $\mathcal{F}$ which is zero modulo some integer N, that is, $\sum_{f \in \mathcal{F}} c_f f \equiv 0 \pmod{N}$. Thus, over the integers, it is $\sum_{f \in \mathcal{F}} c_f f = N f_N$, where f_N is a polynomial comprising only monomials of M. Moreover, $f_N \notin \mathcal{F}$. (If it was, the polynomials in $\mathcal{F}$ would be linearly dependent over the integers. This is impossible due to the construction of $\mathcal{F}$. Namely, the polynomials can be enumerated in such a way that each polynomial introduces a new monomial.) As I is a prime ideal and $N \notin I$ by construction, it is $f_N \in I$. That is, $f_N = \sum_{i=1}^{r} f_i q_i$ with polynomials $f_i \in \mathbb{Z}[x_1, x_2, x_3]$. This, combined with the facts that f_N only contains monomials of M and $f_N \notin \mathcal{F}$, implies that $(S_1, \ldots, S_r, M)$ used to define $\mathcal{F}$ is not admissible, a contradiction to the preconditions. Consequently, no modular linear combination of polynomials of $\mathcal{F}$ equal to zero exists. Hence, $\mathcal{F}$ is determinant preserving.
Moreover, the shift polynomial set $\mathcal{F}$ corresponds to an upper triangular matrix. Consequently, bounds can be calculated directly.
The drawback of this method is, however, that the ideal I is not necessarily prime. If it is not, a prime ideal has to be constructed from I by decomposition and taking the radical. For details, compare [Bau08].
Altogether, Aurélie Bauer and Antoine Joux construct shift polynomial sets in a way to provably determine small solutions. They do not consider, however, if they could get a better bound with a less restrictive choice of shift polynomials. In case of some specific systems of polynomials a heuristic approach might work better. Then we can construct shifts in a way that a lot of monomials reappear. Such methods are, of course, specialized with respect to the system of polynomial equations under consideration and cannot be used for general analyses. If searching for a general method to be applicable to any system, we refer the reader to the approach pursued by A. Bauer and A. Joux [BJ07, Bau08].
Here we restrict our analysis to a special case. In the following section we will return to the example of implicit factoring introduced in Section 4.2 and analyze it over the integers.

6.1 Analyzing the Problem of Implicit Factoring

The problem of implicit factoring can be analyzed in various manners. Before elaborating on its analyses, let us briefly recall the problem. The goal is to factor an n-bit RSA modulus $N_0 = p_0 q_0$, where p_0 and q_0 denote prime numbers. As this problem is difficult in general, we make use of an restricted oracle that on the i-th query, $i = 1, \ldots, k$, outputs another different RSA modulus $N_i = p_i q_i$ such that p_0 and p_i share t bits. The non-shared bits of

p_0 and p_i are subsequent. Moreover, we assume q_0 and q_i to be α-bit numbers.
In Section 4.2 we have shown that we can determine the vector $\mathbf{q} = (q_0, q_1, \ldots, q_k)$ by a heuristic lattice attack using a $(k+1)$-dimensional lattice L in polynomial time if $t > \frac{k+1}{k}\alpha$. The condition there is, however, that the factors p_0 and p_i share their least significant bits. The case of shared most significant bits has been treated in Section 4.3. In that heuristic lattice attack a lattice of size quadratic in the number of oracle queries k is used. The bound obtained is (disregarding a small constant) the same as in the case of shared least significant bits. Both analyses use lattices in which we can extract the factors $\mathbf{q} = (q_0, q_1, \ldots, q_k)$ from a short vector.
Using the more advanced technique by Coppersmith and regarding integer equations, we can partly obtain better bounds. Furthermore, Coppersmith's method allows for a more general analysis, independent of the position of the subsequent non-shared bits. We start with a system of equations we get by putting k oracle queries. This gives us $k+1$ different RSA moduli

$$\begin{aligned} N_0 &= \left(p_l + 2^{t_p}\tilde{p}_0 + 2^{n-\alpha-t+t_p}p_m\right) q_0 \\ &\vdots \\ N_k &= \left(p_l + 2^{t_p}\tilde{p}_k + 2^{n-\alpha-t+t_p}p_m\right) q_k \end{aligned} \tag{6.5}$$

with α-bit q_i.
This is equivalent to

$$\begin{aligned} 2^{t_p}\tilde{p}_0 q_0 - N_0 &= -\left(p_l + 2^{n-\alpha-t+t_p}p_m\right) q_0 \\ &\vdots \\ 2^{t_p}\tilde{p}_k q_k - N_k &= -\left(p_l + 2^{n-\alpha-t+t_p}p_m\right) q_k\,. \end{aligned}$$

We transform the system of equations into a system of k integer equations. For $i = 1, \ldots, k$ we multiply the i-th equation with q_0, the 0-th equation with q_i and combine these two equations. Then we obtain

$$\begin{aligned} 2^{t_p}\tilde{p}_0 q_0 q_1 - N_0 q_1 &= 2^{t_p}\tilde{p}_1 q_0 q_1 - N_1 q_0 \\ &\vdots \\ 2^{t_p}\tilde{p}_0 q_0 q_k - N_0 q_k &= 2^{t_p}\tilde{p}_k q_0 q_k - N_k q_0\,. \end{aligned}$$

Hence, we derive the following system of equations:

$$\begin{aligned} 2^{t_p}\underbrace{(\tilde{p}_1 - \tilde{p}_0)}_{y_1} q_0 q_1 + N_0 \underbrace{q_1}_{x_1} - N_1 \underbrace{q_0}_{x_0} &= 0 \\ &\vdots \\ 2^{t_p}\underbrace{(\tilde{p}_k - \tilde{p}_0)}_{y_k} q_0 q_k + N_0 \underbrace{q_k}_{x_k} - N_k \underbrace{q_0}_{x_0} &= 0\,. \end{aligned}$$

Replacing the values $q_0, \dots, q_k, \tilde{p}_1 - \tilde{p}_0, \dots, \tilde{p}_k - \tilde{p}_0$ by unknowns $x_0, \dots, x_k, y_1, \dots, y_k$, we define the system of polynomial equations

$$\begin{array}{rcl} f_1(x_0, \dots, x_k, y_1, \dots, y_k) & := & 2^{t_p} x_0 x_1 y_1 - N_1 x_0 + N_0 x_1 = 0 \\ & & \vdots \\ f_k(x_0, \dots, x_k, y_1, \dots, y_k) & := & 2^{t_p} x_0 x_k y_k - N_k x_0 + N_0 x_k = 0\,. \end{array} \tag{6.6}$$

Our goal is to solve system (6.6) and determine its solutions applying Coppersmith's method. Thus, if the values $q_0, \dots, q_k, \tilde{p}_1 - \tilde{p}_0, \dots, \tilde{p}_k - \tilde{p}_0$ are small enough, we obtain them by this method. This implies that we can factor all N_i for any position of the bits $\tilde{p}_i$ in p_i. In what follows we will write $f_i(x_0, x_i, y_i)$ instead of $f_i(x_0, \dots, x_k, y_1, \dots, y_k)$ as the polynomial only depends on the variables x_0, x_i, y_i.

6.1.1 The Case of One Equation

First, let us have a look at the equation $f_1(x_0, x_1, y_1) := 2^{t_p} x_0 x_1 y_1 + N_0 x_1 - N_1 x_0 = 0$, which has been determined using the two moduli N_0 and N_1. By construction of f_1, we know that $f_1(q_0, q_1, \tilde{p}_1 - \tilde{p}_0) = 0$.
Let again $X_i = 2^{\alpha}$ and $Y_i = 2^{n-\alpha-t}$ denote the upper bounds on $|q_i|$ and $|\tilde{p}_i - \tilde{p}_0|$, respectively. Let us recall Coppersmith's method presented in Section 2.3. We apply it to find sufficiently small solutions of $f_1(x_0, x_1, y_1) = 0$. For simplicity we only use the polynomial f_1 itself to construct a lattice basis. We do not define any larger shift polynomial sets $\mathcal{F}$ yet. We set

$$\mathbf{B} := \begin{pmatrix} (X_0 X_1 Y_1)^{-1} & 0 & 2^{t_p} \\ 0 & (X_0)^{-1} & -N_1 \\ 0 & 0 & N_0 \end{pmatrix} \begin{array}{c} x_0 x_1 y_1 \\ x_0 \\ x_1 \end{array}$$

The matrix B is a basis matrix of a lattice L containing the vector $(q_0 q_1 (\tilde{p}_1 - \tilde{p}_0), q_0, q_1)\mathbf{B} = (\frac{q_0 q_1 (\tilde{p}_1 - \tilde{p}_0)}{X_0 X_1 Y_1}, \frac{q_0}{X_0}, 0) =: \mathbf{t}$.
There are two properties of the vector $\mathbf{t}$ which we will use. First, the norm of the vector is particularly small, i. e. it holds that $||\mathbf{t}|| \leq \sqrt{\left(\frac{q_0 q_1 (\tilde{p}_1 - \tilde{p}_0)}{X_0 X_1 Y_1}\right)^2 + \left(\frac{q_0}{X_0}\right)^2 + 0} \leq \sqrt{2}$.
Second, the last entry is 0. This implies that $\mathbf{t}$ is also a vector in a sublattice L_S of L of vectors having 0 in their last component. We can determine a basis $\mathbf{B_S}$ of this sublattice via unimodular transformations on $\mathbf{B}$. Furthermore, as the greatest common divisor of the coefficients of f_1 equals one, there is a unimodular matrix $\mathbf{T}$ such that

$$\mathbf{TB} = \begin{pmatrix} \multicolumn{2}{c}{\mathbf{B_S}} & \begin{array}{c} 0 \\ 0 \end{array} \\ * & * & 1 \end{pmatrix}.$$

Thus, it is $\det(\mathbf{B}) = \det(\mathbf{B_S})$. We will apply lattice basis reduction to $\mathbf{B_S}$ and then orthogonalize the reduced basis. Let us denote the result by $\begin{pmatrix} \mathbf{b}_1^* \\ \mathbf{b}_2^* \end{pmatrix}$. By Lemma 2.3.10

we get that $\mathbf{t}$ is a multiple of $\mathbf{b}_1^*$ if

$$||\mathbf{t}|| \;<\; \det(L_S)^{\frac{1}{2}} 2^{-\frac{1}{4}} \,. \tag{6.7}$$

The determinant of the sublattice is $\det(L_S) = \det(L) = \det(\mathbf{B}) = X_0^{-2} X_1^{-1} Y_1^{-1} N_0$. Condition (6.7) is fulfilled if

$$\begin{aligned} 2^{\frac{1}{2}} &\;<\; 2^{\frac{1}{2}(-3\alpha-(n-\alpha-t)+n)-\frac{1}{4}} \\ \Leftrightarrow \quad 2\alpha + \frac{6}{4} &\;<\; t\,. \end{aligned}$$

As α and t are integers as they describe numbers of bits, we require

$$2\alpha + 1 \;<\; t\,.$$

This is essentially the bound we got via the modular analysis (Theorem 4.2.1). A result like this was to be expected. Both during the modular analysis as well as in this simple lattice construction the problem was linearized. Thus, algebraic relations between different monomials were not taken into account.
Hence, we conclude, if $2\alpha + 1 < t$, the vector $\mathbf{t}$ is orthogonal to $\mathbf{b}_2^*$ and we get another equation in the unknowns, namely,

$$f_n(x_0, x_1, y_1) := \frac{x_0 x_1 y_1}{X_0 X_1 Y_1} (\mathbf{b}_2^*)_1 + \frac{x_0}{X_0} (\mathbf{b}_2^*)_2 = 0\,.$$

However, we are not yet finished. We have two equations in three unknowns which does not necessarily imply that we can solve the system of equations. As the greatest common divisor of both equations is one, we can eliminate one of the variables. That is, we get one equation in two unknowns.
Let us consider an example of two 256-bit RSA-moduli with 140 shared bits of the larger 196-bit factor.

Example 6.1.1
Let

$$\begin{aligned} N_0 \;&=\; (2^{56} \cdot 812673540453457095612765286246825237797002 + 66763130916719055) \\ &\quad \cdot 55353788004451507 \end{aligned}$$

$$= 3241479081396338383767602816098777307849166574844502255653955136311902508989\,,$$

$$\begin{aligned} N_1 \;&=\; (2^{56} \cdot 812673540453457095612765286246825237797002 + 71789524880268405) \\ &\quad \cdot 224075567809535399 \end{aligned}$$

$$= 13121708412226537409254487715048399315004981583420160875601531613923194589523\,.$$

The polynomial to be analyzed is then $f_1(x_0, x_1, y_1) := x_0x_1y_1 + N_0x_1 - N_1x_0$ *with root*

$$(q_0, q_1, \tilde{p}_1 - \tilde{p}_0) \quad = \quad (55353788004451507, 224075567809535399, 717895248880268405 - 667631309167190055)\,.$$

Applying Coppersmith's method as described above, we get as second polynomial the polynomial:

$$\begin{aligned} g_1(x_0, x_1, y_1) &= 55353788004451507 \cdot x_0x_1y_1 \\ &- 72633626570661573657492746789045172226708292440208378058740748 35 \\ &\ 1126870878107948573381422576 2 \cdot x_0 \\ &+ 17942814589247712569252389764024487646697692243796043077707715 96 \\ &\ 79479524226168234746077644916 \cdot x_1\,. \end{aligned}$$

The polynomial g_1 *is by construction not a multiple of* $f_1(x_0, x_1, y_1)$*, but shares the same roots. In order to recover a solution, we compute a Groebner basis of* $\{f_1, g_1\}$ *with respect to lexicographic ordering. The Groebner basis is*

$$\{h_1, h_2, h_3\} := \{x_0x_1y_1 - 2782299458851654753512976763704 50x_1,$$

$$224075567809535399x_0 - 55353788004451507x_1,$$

$$55353788004451507x_1^2y_1 - 623445331058347610340623922880285154034751125595 50x_1\}\,.$$

Having a look at the second polynomial of the Groebner basis, namely, $h_2(x_0, x_1, y_1) = 224075567809535399x_0 - 55353788004451507x_1$*, we remark that* $h_2(x_0, x_1, y_1) = q_1x_0 - q_0x_1$*. Thus, we get both factorizations by using the coefficients of* $h_2(x_0, x_1, y_1)$*.*
Remark that in this example we do not even need to compute a Groebner basis as the value q_0 *is also a coefficient of the polynomial* g_1*.*

Note that the variety $V(h_1, h_2, h_3)$ is not zero dimensional. Therefore, we cannot apply the folklore heuristic from [JM06]. Nevertheless, we have seen in Example 6.1.1 that we can determine the solution $(q_0, q_1, \tilde{p}_1 - \tilde{p}_0)$.
Our experiments showed that similar observations generally hold in this setting. A description of the experiments will be given at the end of this section. Based on these observations, we introduce the following heuristic.

Assumption 6.1.2
Let $\mathcal{F}$ *denote a shift polynomial set which is constructed with respect to a polynomial* $f \in \mathbb{Z}[x_0, x_1, y_1]$*. Assume* $\{f, g_1, \ldots, g_\lambda\}$ *is the set of polynomials in* $\mathbb{Z}[x_0, x_1, y_1]$ *we get by applying Coppersmith's method with the shift polynomial set* $\mathcal{F}$*. Let* $GB := \{h_1, \ldots, h_\kappa\}$ *be a Groebner basis of* $\{f, g_1, \ldots, g_\lambda\}$*. Then* GB *contains a polynomial* $h(x_0, x_1, y_1)$ *such that the factors* q_0 *and* q_1 *can be determined as* $\gcd(c, N_i)$*,* $i = 0, 1$*, for appropriate coefficients* c *of* h*.*

A similar heuristic is also used by Santanu Sarkar and Subhamoy Maitra in [SM09] who did related analyses independently. We will elaborate on this in what follows.

Now, let us consider which shifts to use to define $\mathcal{F}$. The polynomial $f_1(x_0, x_1, y_1) = 2^{t_p} x_0 x_1 y_1 + N_0 x_1 - N_1 x_0$ we regard here does not have any constant term. To apply the folklore techniques of analysis presented in [JM06] we would have to substitute e. g. x_0 by $x_0 - 1$ to introduce a constant term. This is the approach Santanu Sarkar and Subhamoy Maitra followed. We will not directly define a standard shift polynomial set, but use a different approach.
For $l \in \mathbb{N}$ we denote by $M_l(f_1)$ the set of monomials occurring in f_1^l. For any monomial $m \in M_l(f_1)$ multiples $mf_1(x_0, x_1, y_1)$ do not contain monomials occurring in any power f_1^λ for $\lambda \leq l$. This is because x_0 and x_1 are of the same total degree, namely one, and $x_0x_1y_1$ contains another variable y_1, i. e. has degree one in this variable. Therefore, we can regard the shift polynomial sets separately as there are no monomials which occur in two shift polynomial sets with different indices l. Furthermore, by increasing the number of shifts, we get higher powers of N_0 in the determinant but also higher powers of X_0^{-1}, X_1^{-1} and Y_1^{-1}. Their ratio, however, remains constant. Thus, we suppose that using such polynomials $mf_1(x_0, x_1, y_1)$, with $m \in M_l(f_1)$, as shift polynomials to build a lattice and to apply Coppersmith's method results in the same bound for any l. This intuition proves correct as can be seen in the proof of the following theorem. That is, the condition is the same using the simplest lattice construction as well as sets of standard shifts. The reason to look at this shift polynomial set anyway is that it can be used for extensions. This is also the motivation to give the proof with respect to any $l \in \mathbb{N}$, otherwise a fixed value of l would be sufficient.

Theorem 6.1.3
Suppose Assumption 6.1.2 holds. Let $N_0, N_1 \in \mathbb{N}$ be of size 2^n and $f_1(x_0, x_1, y_1) = 2^{t_p} x_0 x_1 y_1 + N_0 x_1 - N_1 x_0 \in \mathbb{Z}[x_0, x_1, y_1]$. Let $X_i = 2^\alpha$, $i = 0, 1$, $Y_1 = 2^{n-t-\alpha}$ be bounds on the absolute value of the solutions q_0 and q_1, $\tilde{p}_1 - \tilde{p}_0$, respectively. Then in time polynomial in n we can determine all integer solutions $(q_0, q_1, \tilde{p}_1 - \tilde{p}_0)$ such that $|q_i| \leq X_i$ and $|\tilde{p}_1 - \tilde{p}_0| \leq Y_1$ if

$$2\alpha + 1 < t\,.$$

Proof: Let $l \in \mathbb{N}$. Let $M_l(f_1) = \{x_0^i x_1^j y_1^k \mid k = 0, \ldots, l; i = k, \ldots, l \text{ and } j = l + k - i\}$ be the set of all shift monomials. We take all polynomials $mf_1(x_0, x_1, y_1)$ for $m \in M_l(f_1)$ to construct the lattice. That is, we set $\mathcal{F} := \{mf_1(x_0, x_1, y_1) \mid m \in M_l(f_1)\}$. Let $n_l := |M_l(f_1)|$ be the number of monomials used as shifts. Then

$$n_l = \sum_{k=0}^{l} \sum_{i=k}^{l} 1 = \sum_{k=0}^{l} (l - k + 1) = \frac{1}{2}l^2 + \frac{3}{2}l + 1\,.$$

Let $M_l(f_1) = \{m_1, \ldots, m_{n_l}\}$ be an enumeration of the monomials of $M_l(f_1)$ such that $m_i >_{\text{grlex}} m_j$ if $i < j$, and let $g_{1i}(x_0, x_1, y_1) := m_i f_1(x_0, x_1, y_1)$. Then for every polynomial g_{1i} the monomial $x_1 m_i$ is not contained in the set of monomials of all g_{1j} with $j < i$.

$$\mathbf{B} = \begin{array}{c} \left(\begin{array}{cccccc} (X_0X_1Y_1)^{-2} & 0 & 0 & 2^{t_p} & 0 & 0 \\ 0 & (X_0)^{-2}(X_1Y_1)^{-1} & 0 & -N_1 & 2^{t_p} & 0 \\ 0 & 0 & (X_0)^{-2} & 0 & -N_1 & 0 \\ 0 & 0 & 0 & N_0 & 0 & 2^{t_p} \\ 0 & 0 & 0 & 0 & N_0 & -N_1 \\ 0 & 0 & 0 & 0 & 0 & N_0 \end{array}\right) \end{array} \begin{array}{c} x_0^2x_1^2y_1^2 \\ x_0^2x_1y_1 \\ x_0^2 \\ x_0x_1^2y_1 \\ x_0x_1 \\ x_1^2 \end{array} .$$

Figure 6.1: A lattice basis matrix in the case of $l = 1$.

This can be proven by contradiction. Assume that there exists $j < i$ such that x_1m_i is a monomial of g_{1j}. Then $x_1m_i \in \{x_1m_j, x_0m_j, x_0x_1y_1m_j\}$. If $x_1m_i = x_1m_j$, then $m_i = m_j$ and, consequently, $i = j$, which is a contradiction. If $x_1m_i = x_0m_j$, then the degree of x_0 in m_i is positive and $m_j = m_ix_1x_0^{-1}$. This implies that $\deg(m_i) = \deg(m_j)$ and due to the ordering $m_i >_{\text{grlex}} m_j$. Therefore, $i < j$, a contradiction. If $x_1m_i = x_0x_1y_1m_j$, then $\deg(m_i) > \deg(m_j)$ and, thus, we again get the contradiction $i < j$.
This observation can be used to build an upper triangular basis matrix $\mathbf{B}$ of a lattice L. Such a lattice can then be analyzed easier as e. g. the determinant computation is simply the multiplication of the diagonal elements of $\mathbf{B}$.
The idea of the construction of $\mathbf{B}$ is to let the last rows correspond to the monomials $x_1m_i, i = 1, \ldots, n_l$, and the first rows correspond to all other monomials in any order. An example of a basis matrix $\mathbf{B}$ in the case of $l = 1$ is given in Figure 6.1.
Now we describe the construction more formally. Let r_l be the number of monomials occurring in the set of shift polynomials and $m_{n_l+1}, \ldots, m_{r_l}$ be an enumeration of all the monomials which are not contained in $x_1m_1, \ldots, x_1m_{n_l}$. The monomials $m_{n_l+1}, \ldots, m_{r_l}$ can be ordered in any way. For consistency we assume that they are ordered such that $m_i >_{\text{grlex}} m_j$ if $i < j$. We set $\mathbf{x} := (m_{n_l+1}, \ldots, m_{r_l}, x_1m_1, \ldots, x_1m_{n_l})$.
Furthermore, let M_i denote the evaluation of m_i in the values (X_0, X_1, Y_1). Let $\mathbf{D} := \text{Diag}(M_{n_l+1}^{-1}, \ldots, M_{r_l}^{-1})$. Let $\mathbf{f}$ denote the coefficient vector of $f(x_0, x_1, y_1)$ ordered such that $\mathbf{f}\mathbf{x}^T = f(x_0, x_1, y_1)$. Let $\mathbf{F}$ be the matrix consisting of the coefficient vectors $(\mathbf{g_{1i}})^T$ for $i = 1, \ldots, n_l$. That is, $\mathbf{F}$ denotes the matrix induced by $\mathcal{F}$. Using these definitions, we define a lattice L via the basis matrix

$$\mathbf{B} = \left(\begin{array}{cc} \mathbf{D} & \\ & \mathbf{F} \\ \mathbf{0} & \end{array}\right) \begin{array}{c} m_{n_l+1} \\ \vdots \\ m_{r_l} \\ x_1m_1 \\ \vdots \\ x_1m_{n_l} \end{array} .$$

We apply Coppersmith's algorithm to this lattice. Then we get a second polynomial, coprime to f_1, on condition that $\det(L)^{\frac{1}{r_l-n_l}} 2^{-\frac{r_l-n_l-1}{4}} > \sqrt{r_l - n_l}$.
We calculate the determinant of our lattice L. Recall that $\mathbf{B}$ is an upper triangular matrix

by construction. Thus, the matrix $\mathbf{F}$ can be divided into an $(r_l - n_l) \times n_l$ matrix $\mathbf{F_1}$ and an upper triangular $n_l \times n_l$ matrix $\mathbf{F_2}$. More precisely, $\mathbf{F} = \begin{pmatrix} \mathbf{F_1} \\ \mathbf{F_2} \end{pmatrix}$. It holds that $\det(L) = P_1 \cdot P_2$, where P_1 denotes the product of the diagonal entries of $\mathbf{D}$ and P_2 denotes the product of the diagonal entries of $\mathbf{F_2}$. The elements on the diagonal of $\mathbf{F_2}$ correspond to monomials of type $x_1 m_i$, $i = 1, \ldots, n_l$. Hence, all values on the diagonal of $\mathbf{F}$ are equal to N_0. Their number is $n_l = \frac{1}{2}l^2 + \frac{3}{2}l + 1$. Consequently, we have $P_2 = N_0^{\frac{1}{2}l^2+\frac{3}{2}l+1} \approx 2^{n\left(\frac{1}{2}l^2+\frac{3}{2}l+1\right)}$.

In order to evaluate P_1, we need to describe the monomials $m_{n_l+1}, \ldots, m_{r_l}$ explicitly. The set of all monomials occurring in the shift polynomials is exactly

$$M_{l+1}(f_1) = \{x_0^i x_1^j y_1^k \mid k = 0, \ldots, l+1; i = k, \ldots, l+1 \text{ and } j = l+1+k-i\}.$$

The set $M := \{x_1 m_1, \ldots, x_1 m_{n_l}\}$ can be described as

$$M = \{x_0^i x_1^j y_1^k \mid k = 0, \ldots, l; i = k, \ldots, l \text{ and } j = l+1+k-i\}\,.$$

The powers of any of X_0^{-1}, X_1^{-1} and Y_1^{-1} occurring on the diagonal of $\mathbf{D}$ correspond to the powers of the variables x_0, x_1, y_1 occurring in monomials in the set $M_{l+1}(f_1) \setminus M$. Let s_0 and s_1 denote the powers of X_0^{-1} and X_1^{-1}, respectively, and let u denote the power of Y_1^{-1}. Then we have

$$\begin{aligned} s_0 &= \sum_{k=0}^{l+1}\sum_{i=k}^{l+1} i - \sum_{k=0}^{l}\sum_{i=k}^{l} i \\ &= l^2 + 3l + 2\,, \end{aligned} \tag{6.8}$$

$$\begin{aligned} s_1 &= \sum_{k=0}^{l+1}\sum_{i=k}^{l+1} (l+1+k-i) - \sum_{k=0}^{l}\sum_{i=k}^{l} (l+k-i+1) \\ &= \frac{1}{2}l^2 + \frac{3}{2}l + 1\,, \end{aligned} \tag{6.9}$$

$$\begin{aligned} u &= \sum_{k=0}^{l+1}\sum_{i=k}^{l+1} k - \sum_{k=0}^{l}\sum_{i=k}^{l} k \\ &= \frac{1}{2}l^2 + \frac{3}{2}l + 1\,. \end{aligned} \tag{6.10}$$

This gives

$$P_1 = X_0^{-\left(l^2+3l+2\right)} X_1^{-\left(\frac{1}{2}l^2+\frac{3}{2}l+1\right)} Y_1^{-\left(\frac{1}{2}l^2+\frac{3}{2}l+1\right)}.$$

Using the calculations of P_1, P_2 and $X_0 = X_1 = 2^\alpha$, $Y_1 = 2^{n-\alpha-t}$ as well as $N_0 \geq 2^{n-1}$, we obtain

$$\det(L) \geq 2^{-\alpha\left(\frac{3}{2}l^2+\frac{9}{2}l+3\right)-(n-\alpha-t)\left(\frac{1}{2}l^2+\frac{3}{2}l+1\right)+n\left(\frac{1}{2}l^2+\frac{3}{2}l+1\right)-\left(\frac{1}{2}l^2+\frac{3}{2}l+1\right)}$$

$$= 2^{t\left(\frac{1}{2}l^2+\frac{3}{2}l+1\right)-\alpha\left(l^2+3l+2\right)-\left(\frac{1}{2}l^2+\frac{3}{2}l+1\right)}\,.$$

In order to apply the condition given in Lemma 2.3.10, we have to determine the dimension r of the sublattice L_S in Coppersmith's method. As the last n_l entries of the target vector are zeros, it is $r = r_l - n_l$. We have $r_l = |M_{l+1}(f_1)| = \sum_{k=0}^{l+1}\sum_{i=k}^{l+1} 1 = \frac{1}{2}l^2 + \frac{5}{2}l + 3$. Consequently,

$$r = \left(\frac{1}{2}l^2 + \frac{5}{2}l + 3\right) - \left(\frac{1}{2}l^2 + \frac{3}{2}l + 1\right) = l + 2\,.$$

All components of the target vector $\mathbf{t}$ are smaller than 1 so that $||\mathbf{t}|| \leq \sqrt{l+2}$. Therefore, we get the following condition on the existence of a second equation:

$$\sqrt{l+2} < \left(2^{t\left(\frac{1}{2}l^2+\frac{3}{2}l+1\right)-\alpha\left(l^2+3l+2\right)-\left(\frac{1}{2}l^2+\frac{3}{2}l+1\right)}\right)^{\frac{1}{l+2}} \cdot 2^{-\frac{l+1}{4}}$$

$$\Leftrightarrow \frac{l+2}{2}\log(l+2) + \frac{(l+1)(l+2)}{4} + \left(\frac{1}{2}l^2 + \frac{3}{2}l + 1\right) < t\left(\frac{1}{2}l^2 + \frac{3}{2}l + 1\right) - \alpha\left(l^2 + 3l + 2\right)$$

$$\Leftrightarrow \quad 2\alpha + \frac{3}{2} + \frac{\log(l+2)}{l+1} < t\,.$$

If $l = 0, \ldots, 4$, our condition is $2\alpha + 2 < t$, for any $l > 4$, it is $2\alpha + 1 < t$.
On the heuristic Assumption 6.1.2 this implies that we are able to determine the solution $(q_0, q_1, \tilde{p}_1 - \tilde{p}_0)$. As we can use the construction with $l = 5$, we have a fixed number of components in the basis vectors and a fixed lattice dimension so that the running time of the LLL algorithm 2.3.5 only depends on the size of the largest values in the lattice, which is determined by n. Thus, the attack works in time polynomial in n. This proves the theorem. ■

Remark that we get essentially the same lower bound on t with this method as we have obtained with the modular approach (Theorem 4.2.1). Ignoring the bit which has to be added, this bound is illustrated by the function $g(\alpha) = 2\alpha$ in Figure 6.2 in comparison to further bounds. This way, it corresponds to the result of the modular analysis.

Sometimes better bounds can be achieved via so called extra shifts, i. e. instead of regarding polynomials mf_1 such that $m \in M_l(f_1)$, we take different monomials m. The question now is how to choose the monomials m appropriately. As we would like to have a large determinant, a "good" polynomial in the shift polynomial set is a polynomial which introduces a large term on the determinant. Thus, we would like to have a polynomial with a new monomial with a large coefficient, and as few other new monomials as possible. Therefore, we take shifts which transform monomials of one polynomial into other monomials which already exist. Take a look at the monomials of $f_1(x_0, x_1, y_1)$. They are x_0, x_1 and $x_0x_1y_1$. Multiplying x_0 by x_1y_1 results in $x_0x_1y_1$, and, analogously, $x_1 \cdot (x_0y_1) = x_0x_1y_1$. Thus, a promising approach seems to be to choose $m \in \{(x_0y_1)^a(x_1y_1)^b, a, b = 0, \ldots, l\} =: E_l(f_1)$ as the following example shows.

Example 6.1.4
Let $l = 1$ and use $mf_1(x_0, x_1, y_1)$ with $m \in E_1(f_1)$ as set of shift polynomials. That is, $\mathcal{F} := \{x_0x_1y_1^2f_1, x_0y_1f_1, x_1y_1f_1, f_1\}$. Then we define $\mathbf{B}$ as follows:

$$\mathbf{B} := \left(\begin{array}{cccccccc}
(X_0^2X_1^2Y_1^3)^{-1} & 0 & 0 & 0 & 2^{t_p} & 0 & 0 & 0 \\
0 & (X_0^2X_1Y_1^2)^{-1} & 0 & 0 & -N_1 & 2^{t_p} & 0 & 0 \\
0 & 0 & (X_0^2Y_1)^{-1} & 0 & 0 & -N_1 & 0 & 0 \\
0 & 0 & 0 & (X_0)^{-1} & 0 & 0 & 0 & -N_1 \\
0 & 0 & 0 & 0 & N_0 & 0 & 2^{t_p} & 0 \\
0 & 0 & 0 & 0 & 0 & N_0 & -N_1 & 2^{t_p} \\
0 & 0 & 0 & 0 & 0 & 0 & N_0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & N_0
\end{array}\right)
\begin{array}{c}
x_0^2x_1^2y_1^3 \\ x_0^2x_1y_1^2 \\ x_0^2y_1 \\ x_0 \\ x_0x_1^2y_1^2 \\ x_0x_1y_1 \\ x_1^2y_1 \\ x_1
\end{array}$$

The matrix $\mathbf{B}$ is a basis matrix of a lattice L with determinant $\det(L) = X_0^{-7}X_1^{-3}Y_1^{-6}N_0^4 \geq 2^{-10\alpha}2^{-6(n-\alpha-t)}2^{4(n-1)}$ for $X_0 = X_1 = 2^{\alpha}$, $Y_1 = 2^{(n-\alpha-t)}$ and $N_0, N_1 \geq 2^{n-1}$.
Let again $\mathbf{t} := \mathbf{sB} = (\frac{q_0^2q_1^2(\tilde{p}_1-\tilde{p}_0)^3}{X_0^2X_1^2Y_1^3}, \frac{q_0^2q_1(\tilde{p}_1-\tilde{p}_0)^2}{X_0^2X_1Y_1^2}, \frac{q_0^2(\tilde{p}_1-\tilde{p}_0)}{X_0^2Y_1}, \frac{q_0}{X_0}, 0, 0, 0, 0)$ with $\mathbf{s} := (q_0^2q_1^2(\tilde{p}_1 - \tilde{p}_0)^3, q_0^2q_1(\tilde{p}_1 - \tilde{p}_0)^2, q_0^2(\tilde{p}_1 - \tilde{p}_0), q_0, q_0q_1^2(\tilde{p}_1 - \tilde{p}_0)^2, q_0q_1(\tilde{p}_1 - \tilde{p}_0), q_1^2(\tilde{p}_1 - \tilde{p}_0), q_1)$ denote our target vector. It is $||\mathbf{t}|| \leq \sqrt{4} = 2$. Thus, by Lemma 2.3.10, we get another equation in the unknowns not being a multiple of the original one if

$$\begin{aligned}
||\mathbf{t}|| \leq 2 &< \left(2^{-10\alpha}2^{-6(n-\alpha-t)}2^{4(n-1)}\right)^{\frac{1}{4}} 2^{-\frac{3}{4}} \\
\Leftrightarrow 10\alpha + 6(n - \alpha - t) + 7 &< 4(n-1) \\
\Leftrightarrow \frac{4\alpha + 2n + 11}{6} &< t\,.
\end{aligned}$$

A natural upper bound on t is given by the fact that the number of shared bits of one factor has to be smaller than the total number of bits of this factor, which implies that we have

$$\begin{aligned}
t \leq n - \alpha \ \Rightarrow\ & \frac{4\alpha + 2n + 11}{6} \leq n - \alpha \\
\Leftrightarrow\ & \alpha \leq \frac{2n}{5} - \frac{11}{10}\,.
\end{aligned}$$

Consequently, the attack described in this example can only be applied to imbalanced moduli with $\alpha \leq \frac{2n}{5} - \frac{11}{10}$. We again have a lattice of constant size. Thus, the running time is only influenced by the size n of the largest value in the matrix. Hence, the attack can be performed in time polynomial in n.

Comparing the lower bound $\frac{4\alpha+2n+11}{6} < t$ of Example 6.1.4 to the lower bound $2\alpha + 1 < t$ of Theorem 6.1.3, we have

$$\begin{aligned}
\frac{4\alpha + 2n + 11}{6} &< 2\alpha + 1 \\
\Leftrightarrow \quad \frac{n}{4} + \frac{5}{8} &< \alpha\,.
\end{aligned}$$

That is, for factors q_i of bitsize α in the range from approximately $\frac{n}{4}$ to $\frac{2n}{5}$ we obtain better results using this simple set of new shift polynomials. The result is illustrated in Figure 6.2 as function e.
Analogously, we can apply Coppersmith's method with the shift polynomial set $\mathcal{F} := \{mf_1(x_0, x_1, y_1) \mid m \in E_l(f_1)\}$ for any $l \in \mathbb{N}$. The analysis is straightforward. We obtain better results by combining the basic approach of Theorem 6.1.3 with further shifts by monomials $m \in E_l(f_1)$. Therefore, we do not give the analysis with respect to this shift polynomial set $\mathcal{F}$ here. We only remark that the bound we obtain regarding the limit $l \to \infty$ only depends on n. Namely, it is $f(\alpha) := \frac{n}{2} < t$. This bound is illustrated in Figure 6.2.
In the following theorem we will give the bound obtained by a combined analysis. That is, we will use shifts in monomials of $M_l(f_1)$ and in monomials of $E_l(f_1)$ as well as combinations of those shifts. Shifts in monomials $m \in E_l(f_1)$ are called extra shifts. Note that in contrast to what is usually done, we do not use extra shifts in single variables here but extra shifts in special monomials.

Theorem 6.1.5
Suppose Assumption 6.1.2 holds. Let $N_0, N_1 \in \mathbb{N}$ be n-bit numbers and $f_1(x_0, x_1, y_1) = 2^{t_p}x_0x_1y_1 + N_0x_1 - N_1x_0 \in \mathbb{Z}[x_0, x_1, y_1]$. Let $X_i = 2^\alpha$, $i = 0, 1$, $Y_1 = 2^{n-t-\alpha}$ be bounds on the solutions q_0 and q_1, $\tilde{p}_1 - \tilde{p}_0$, respectively. Then for all $\epsilon > 0$ there is $l(\epsilon)$ such that we can determine all integer solutions $(q_0, q_1, \tilde{p}_1 - \tilde{p}_0)$ with $|q_i| \le X_i$ and $|\tilde{p}_1 - \tilde{p}_0| \le Y_1$ in time polynomial in $n, l(\epsilon)$ if

$$2\alpha\left(1 - \frac{\alpha}{n}\right) + \frac{3}{2} < t - \epsilon\,.$$

PROOF: Let $l \in \mathbb{N}, \tau \in \mathbb{R}$ such that $\tau l \in \mathbb{N}$. Let $S_{l,\tau l}(f_1) := \{x_0^{i+a}x_1^{j+b}y_1^{a+b+k} \mid k = 0, \ldots, l; i = k, \ldots, l; j = l + k - i; a = 0, \ldots, \tau l \text{ and } b = 0, \ldots, \tau l\}$ be the set of all shift monomials. We take all polynomials $mf_1(x_0, x_1, y_1)$ for $m \in S_{l,\tau l}(f_1)$ to construct the lattice. That is, we set $\mathcal{F} := \{mf_1(x_0, x_1, y_1) \mid m \in S_{l,\tau l}(f_1)\}$. Let $n_l := |S_{l,\tau l}(f_1)|$ be the number of monomials used as shifts. For ease of the analysis we now write $S_{l,\tau l}(f_1) = \{x_0^i x_1^j y_1^k \mid k = 0, \ldots, l + 2\tau l; i = \max\{0, k - \tau l\}, \ldots, \min\{l + \tau l, l + k\} \text{ and } j = l + k - i\}$. Then

$$n_l = \sum_{k=0}^{\tau l}\sum_{i=0}^{k+l} 1 + \sum_{k=\tau l+1}^{2\tau l+l}\sum_{i=k-\tau l}^{l+\tau l} 1 = \tau^2 l^2 + 2\tau l + \frac{3}{2}l + 1 + \frac{1}{2}l^2 + 2\tau l^2\,.$$

Let $S_{l,\tau l}(f_1) = \{m_1, \ldots, m_{n_l}\}$ be an enumeration of the monomials of $S_{l,\tau l}(f_1)$ such that $m_i >_{\text{grlex}} m_j$ if $i < j$, and let $g_{1i}(x_0, x_1, y_1) := m_i f_1(x_0, x_1, y_1)$. Then, as in the proof of Theorem 6.1.3, for every polynomial g_{1i} the monomial $x_1 m_i$ is not contained in the set of monomials of all g_{1j} with $j < i$. We now proceed analogously to that proof and construct an upper triangular basis matrix $\mathbf{B}$ of a lattice L to which we can apply Coppersmith's method.
Recall the notation used: Let r_l be the number of monomials occurring in the set of shift polynomials and $m_{n_l+1}, \ldots, m_{r_l}$ be an enumeration of all these monomials which are not contained in $\{x_1m_1, \ldots, x_1m_{n_l}\}$. Let the monomials $m_{n_l+1}, \ldots, m_{r_l}$ be ordered

such that $m_i >_{\text{grlex}} m_j$ if $i < j$. We set $\mathbf{x} := (m_{n_l+1}, \dots, m_{r_l}, x_1 m_1, \dots, x_1 m_{n_l})$. Furthermore, let M_i denote the evaluation of m_i in the values (X_0, X_1, Y_1). Let $\mathbf{D} := \text{Diag}(M_{n_l+1}^{-1}, \dots, M_{r_l}^{-1})$. Again, let $\mathbf{f}$ denote the coefficient vector of $f(x_0, x_1, y_1)$ ordered such that $\mathbf{f}\mathbf{x}^T = f(x_0, x_1, y_1)$. Let $\mathbf{F}$ be the matrix consisting of the coefficient vectors $(\mathbf{g_{1i}})^T = (\mathbf{m_i f_1})^T$ for $i = 1, \dots, n_l$.
Then we define the lattice L via a basis matrix

$$\mathbf{B} = \begin{array}{c} \left(\begin{array}{cc} \mathbf{D} & \\ & \mathbf{F} \\ \mathbf{0} & \end{array} \right) \end{array} \begin{array}{c} m_{n_l+1} \\ \vdots \\ m_{r_l} \\ x_1 m_1 \\ \vdots \\ x_1 m_{n_l} \end{array}$$

We apply Coppersmith's algorithm to this lattice. By this method we get a coprime second polynomial provided that $\det(L)^{\frac{1}{r_l - n_l}} \cdot 2^{-\frac{r_l - n_l - 1}{4}} > \sqrt{r_l - n_l}$.
Therefore, we determine the determinant of our lattice like in Theorem 6.1.3. It holds that $\det(L) = P_1 \cdot P_2$, where P_1 denotes the product of the diagonal entries of $\mathbf{D}$, and P_2 denotes the product of the diagonal entries of $\mathbf{F_2}$. Again $\mathbf{F_2}$ is the upper triangular $n_l \times n_l$ matrix such that $\mathbf{F} = \begin{pmatrix} \mathbf{F_1} \\ \mathbf{F_2} \end{pmatrix}$. All contributions we derive from diagonal entries of $\mathbf{F_2}$ are equal to N_0. Their number is $n_l = \tau^2 l^2 + 2\tau l + \frac{3}{2} l + 1 + \frac{1}{2} l^2 + 2\tau l^2$. Consequently, we have $P_2 = N_0^{\tau^2 l^2 + 2\tau l + \frac{3}{2} l + 1 + \frac{1}{2} l^2 + 2\tau l^2}$.
In order to evaluate P_1, we need to describe the monomials $m_{n_l+1}, \dots, m_{r_l}$ explicitly. The set of all monomials occurring in the shift polynomials is exactly

$$S_{l+1,\tau l}(f_1) := \{x_0^i x_1^j y_1^k \mid k = 0, \dots, l+1+2\tau l; i = \max\{0, k-\tau l\}, \dots, \min\{l+1+\tau l, l+1+k\}$$

$$\text{and } j = l + 1 + k - i\}.$$

The set $M := \{x_1 m_1, \dots, x_1 m_{n_l}\}$ can be described as

$$M = \{x_0^i x_1^j y_1^k \mid k = 0, \dots, l + 2\tau l; i = \max\{0, k - \tau l\}, \dots, \min\{l + \tau l, l + k\}$$

$$\text{and } j = l + 1 + k - i\}.$$

The number of powers of any of X_0^{-1}, X_1^{-1} and Y_1^{-1} occurring on the diagonal of $\mathbf{D}$ equals the number of powers of the variables x_0, x_1, y_1 occurring in monomials in $S_{l+1,\tau l}(f_1) \setminus M$, respectively. Let s_0 and s_1 denote the powers of X_0^{-1} and X_1^{-1}, respectively, and u denote the power of Y_1^{-1}. Then we have

$$\begin{aligned} s_0 &= \sum_{k=0}^{\tau l} \sum_{i=0}^{l+1+k} i + \sum_{k=\tau l+1}^{2\tau l+l+1} \sum_{i=k-\tau l}^{l+1+\tau l} i \qquad (6.11)\\ &\quad - \left(\sum_{k=0}^{\tau l} \sum_{i=0}^{l+k} i + \sum_{k=\tau l+1}^{2\tau l+l} \sum_{i=k-\tau l}^{l+\tau l} i \right) \\ &= 2 + 3l + l^2 + \frac{7}{2}\tau l + 3\tau l^2 + \frac{3}{2}\tau^2 l^2 , \\ s_1 &= \sum_{k=0}^{\tau l} \sum_{i=0}^{l+1+k} (l+1+k-i) + \sum_{k=\tau l+1}^{2\tau l+l+1} \sum_{i=k-\tau l}^{l+1+\tau l} (l+1+k-i) \qquad (6.12)\\ &\quad - \left(\sum_{k=0}^{\tau l} \sum_{i=0}^{l+k} (l+1+k-i) + \sum_{k=\tau l+1}^{2\tau l+l} \sum_{i=k-\tau l}^{l+\tau l} (l+1+k-i) \right) \\ &= 1 + \frac{3}{2}l + \frac{1}{2}l^2 + \frac{3}{2}\tau l + \tau l^2 + \frac{1}{2}\tau^2 l^2 , \end{aligned}$$

$$\begin{aligned} u &= \sum_{k=0}^{\tau l} \sum_{i=0}^{l+1+k} k + \sum_{k=\tau l+1}^{2\tau l+l+1} \sum_{i=k-\tau l}^{l+1+\tau l} k \qquad (6.13)\\ &\quad - \left(\sum_{k=0}^{\tau l} \sum_{i=0}^{l+k} k + \sum_{k=\tau l+1}^{2\tau l+l} \sum_{i=k-\tau l}^{l+\tau l} k \right) \\ &= 1 + \frac{3}{2}l + \frac{1}{2}l^2 + 3\tau l + 2\tau l^2 + 2\tau^2 l^2 . \end{aligned}$$

This gives

$$P_2 = X_0^{-\left(2+3l+l^2+\frac{7}{2}\tau l+3\tau l^2+\frac{3}{2}\tau^2 l^2\right)} X_1^{-\left(1+\frac{3}{2}l+\frac{1}{2}l^2+\frac{3}{2}\tau l+\tau l^2+\frac{1}{2}\tau^2 l^2\right)} Y_1^{-\left(1+\frac{3}{2}l+\frac{1}{2}l^2+3\tau l+2\tau l^2+2\tau^2 l^2\right)} .$$

Substituting P_1, P_2 and $X_0 = X_1 = 2^{\alpha}$, $Y_1 = 2^{n-\alpha-t}$ as well as $N_0 \geq 2^{n-1}$ in $\det(L)$, we get

$$\begin{aligned} &\det(L) \geq \\ &2^{-\alpha\left(3+\frac{9}{2}l+\frac{3}{2}l^2+5\tau l+4\tau l^2+2\tau^2 l^2\right)-(n-\alpha-t)\left(1+\frac{3}{2}l+\frac{1}{2}l^2+3\tau l+2\tau l^2+2\tau^2 l^2\right)+(n-1)\left(\tau^2 l^2+2\tau l+\frac{3}{2}l+1+\frac{1}{2}l^2+2\tau l^2\right)} \\ &= 2^{t\left(1+\frac{3}{2}l+\frac{1}{2}l^2+3\tau l+2\tau l^2+2\tau^2 l^2\right)-\alpha\left(2\tau l^2+2\tau l+l^2+3l+2\right)-n\left(\tau^2 l^2+\tau l\right)-\left(\tau^2 l^2+2\tau l+\frac{3}{2}l+1+\frac{1}{2}l^2+2\tau l^2\right)} . \end{aligned}$$

In order to apply the condition given in Lemma 2.3.10, we have to determine the dimension r of the sublattice L_S in Coppersmith's method. It is $r_l = |S_{l+1,\tau l}(f_1)| = \sum_{k=0}^{\tau l} \sum_{i=0}^{l+1+k} 1 + \sum_{k=\tau l+1}^{2\tau l+l+1} \sum_{i=k-\tau l}^{l+1+\tau l} 1 = 4\tau l + 3 + \tau^2 l^2 + 2\tau l^2 + \frac{1}{2}l^2 + \frac{5}{2}l$. Consequently,

$$\begin{aligned} r &= r_l - n_l \\ &= \left(4\tau l + 3 + \tau^2 l^2 + 2\tau l^2 + \frac{1}{2}l^2 + \frac{5}{2}l\right) - \left(\tau^2 l^2 + 2\tau l + \frac{3}{2}l + 1 + \frac{1}{2}l^2 + 2\tau l^2\right) \\ &= 2\tau l + l + 2 . \end{aligned}$$

All components of the target vector $\mathbf{t}$ are smaller than or equal to 1 so that $||\mathbf{t}|| \leq \sqrt{2\tau l + l + 2}$. Therefore, we get the following condition on the existence of a second equation:

$$\sqrt{2\tau l + l + 2} < 2^{-\frac{2\tau l + l + 1}{4}}$$

$$\cdot \left(2^{t\left(1+\frac{3}{2}l+\frac{1}{2}l^2+3\tau l+2\tau l^2+2\tau^2 l^2\right)-\alpha\left(2\tau l^2+2\tau l+l^2+3l+2\right)-n\left(\tau^2 l^2+\tau l\right)-\left(\tau^2 l^2+2\tau l+\frac{3}{2}l+1+\frac{1}{2}l^2+2\tau l^2\right)}\right)^{\frac{1}{2\tau l+l+2}}$$

$$\begin{aligned} \Leftrightarrow \quad & \frac{2\tau l + l + 2}{2} \log\left(2\tau l + l + 2\right) + \frac{(2\tau l + l + 1)(2\tau l + l + 2)}{4} \\ & + \left(\tau^2 l^2 + 2\tau l + \frac{3}{2} l + 1 + \frac{1}{2} l^2 + 2\tau l^2\right) \\ < \quad & t \frac{\left(2\tau l + l + 2\right)\left(2\tau l + l + 1\right)}{2} - \alpha\left(2\tau l + l + 2\right)\left(l + 1\right) - n\left(\tau^2 l^2 + \tau l\right) . \end{aligned}$$

As

$$\frac{2\left(\tau^2 l^2 + 2\tau l + \frac{3}{2} l + 1 + \frac{1}{2} l^2 + 2\tau l^2\right)}{4\tau^2 l^2 + 4\tau l^2 + 6\tau l + l^2 + 3l + 2} = 1 - \frac{2\tau^2 l^2 + 2\tau l}{4\tau^2 l^2 + 4\tau l^2 + 6\tau l + l^2 + 3l + 2}$$

we use the slightly stronger constraint

$$\Leftarrow \underbrace{\frac{2(l+1)}{2\tau l + l + 1}\alpha + \frac{2\tau l\left(\tau l + 1\right)}{\left(2\tau l + l + 2\right)\left(2\tau l + l + 1\right)} n + \frac{3}{2} + \frac{\log\left(2\tau l + l + 2\right)}{2\tau l + l + 1}}_{=:h(\alpha,n,l,\tau)} < t \, . \tag{6.14}$$

This implies that for fixed l and τ we can factor the integers N_i if t is greater than $h(\alpha, n, l, \tau)$. Remark that the result of Theorem 6.1.3 is exactly this result in the case of $\tau = 0$.
It is

$$\bar{h}(\alpha, n, \tau) := \lim_{l \to \infty} h(\alpha, n, l, \tau) = \frac{12\tau^2 + 3 + 8\tau\alpha + 12\tau + 4\alpha + 4\tau^2 n}{8\tau^2 + 8\tau + 2} \, .$$

Let $\epsilon > 0$. Then there is $l(\epsilon)$ such that for $l > l(\epsilon)$ condition (6.14) simplifies to

$$\frac{12\tau^2 + 3 + 8\tau\alpha + 12\tau + 4\alpha + 4\tau^2 n}{8\tau^2 + 8\tau + 2} \quad < \quad t - \epsilon \, . \tag{6.15}$$

The optimal value τ_{opt} of τ, such that the lower bound on the number of shared bits $\bar{h}(\alpha, n, \tau_{\text{opt}})$ is as small as possible, is $\tau_{\text{opt}} = \frac{\alpha}{n - 2\alpha}$ for $\alpha \neq \frac{n}{2}$. Substituting this value in the function, we can determine the lower bound on t as

$$\bar{h}(\alpha, n, \tau_{opt}) = \frac{4\alpha n + 3n - 4\alpha^2}{2n} \quad < \quad t - \epsilon \, .$$

This is equivalent to

$$2\alpha\left(1 - \frac{\alpha}{n}\right) + \frac{3}{2} \quad < \quad t - \epsilon \, . \tag{6.16}$$

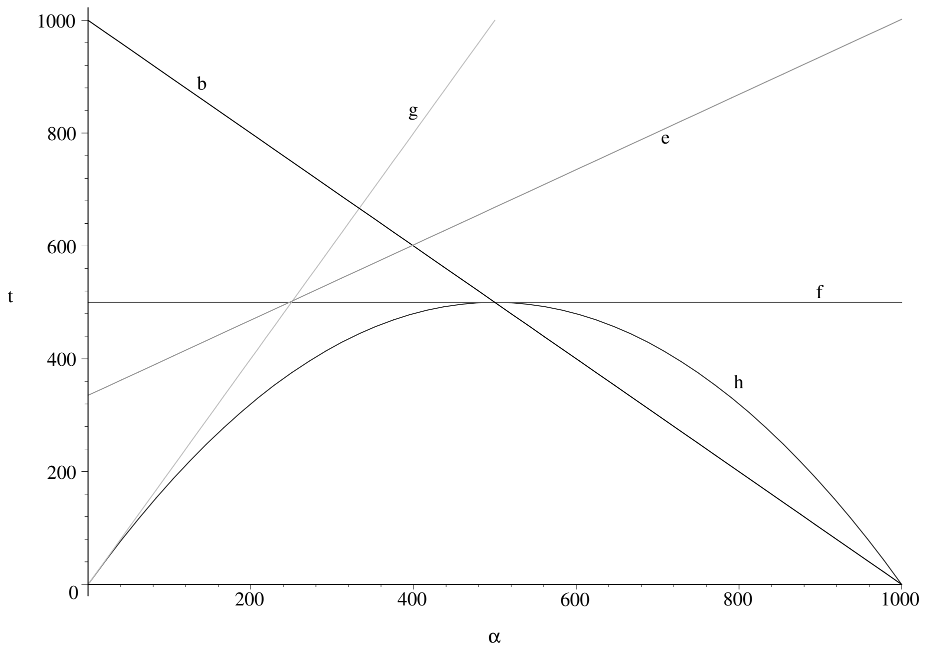

Figure 6.2: $b : t < n - \alpha$, $e : t > \frac{4\alpha+2n+11}{6}$, $f : t > \frac{n}{2}$, $g : t > 2\alpha$, $h : t > 2\alpha\left(1 - \frac{\alpha}{n}\right) + \frac{3}{2}$

Bounds on the number t of shared bits, dependent on the bitsize α of the smaller factor with $n = 1000$.

On the heuristic Assumption 6.1.2 this implies that we are able to determine the solution and proves the bound. The running time of the algorithm is dominated by the running time of the LLL-reduction, which is polynomial in the lattice dimension and the size of the largest values in the lattice. The former is polynomial in $l(\epsilon)$, the latter in n. This concludes the proof. ■

Remark that the condition $\frac{4\alpha n+3n-4\alpha^2}{2n} < t-\epsilon$ implies $\frac{4\alpha n+3n-4\alpha^2}{2n} \leq n-\alpha$, $\alpha \neq \frac{n}{2}$ as $t \leq n-\alpha$ by definition. Thus, $\alpha < \frac{3}{4}n - \frac{\sqrt{n^2+12n}}{4}$. This condition implies that $\alpha < \frac{1}{2}n$. Therefore, Theorem 6.1.5 can only be applied to composite numbers with imbalanced factors.

In order to obtain better bounds, we have used monomials of f_1^l as shift monomials. In contrast to our first analysis, they are useful here as further monomials reoccur in shifted polynomials if the shifts we use are a combination of those shifts and extra shifts.

Furthermore, note that the motivating Example 6.1.4 is not included in our analysis. However, this does not pose any problems as the bound for valid values of α gets better already in the case of $l = \tau = 1$.

In Figure 6.2, a comparison of the various bounds is given.

α	theoretical result of [SM09]	experimental result of [SM09]	theoretical result of modular approach	theoretical bound with $l=4$, $\tau=\frac{1}{2}$ in integer approach	experimental result with $l=4$, $\tau=\frac{1}{2}$ in integer approach	asymptotic bound of integer approach
230	505	434	460	391	390	356
240	521	446	480	402	401	367
250	525	454	500	413	413	377
260	530	472	520	425	423	387
270	534	479	540	436	434	396

Table 6.1: Lower bounds on t for special values of α with $n = 1000$

To verify our heuristic assumptions in practice, we have run experiments on a Core2 Duo 1.66GHz notebook. The attacks were implemented using Magma[1] Version 2.11. As in Section 4.2, we have used the implementation of L^2 by Phong Nguyen and Damien Stehlé [NS05], which is implemented in Magma. We performed experiments using 1000-bit moduli N_i with various bitsizes of the q_i and several values of t_p. We set $l = 4$ and $\tau = \frac{1}{2}$. Then the heuristic was valid for all $t > h(\alpha, n, l, \tau)$ in our experiments. Sometimes an attack was also successful although t was a few bits smaller than the theoretical bound. The bounds obtained for some values of α are given in columns five and six of Table 6.1.

Santanu Sarkar and Subhamoy Maitra independently used an integer approach to analyze the problem of implicit factoring with one oracle call [SM09]. They regard the same polynomial $f_1(x_0, x_1, y_1) = 2^{t_p}x_0x_1y_1 - N_1x_0 + N_0x_1$. However, instead of adapting the shifts to a polynomial without a constant term, they substitute x_0 by $x_0 - 1$ and use the folklore analysis given in [JM06] to determine the roots of $\hat{f}_1(x_0, x_1, y_1) = 2^{t_p}x_0x_1y_1 - 2^{t_p}x_1y_1 - N_1x_0 + N_1 + N_0x_1$. By this, they get the following result: The factorization of N_0 and N_1 can be efficiently determined if the heuristic given in Assumption 6.1.2 holds and

$$-4\frac{\alpha^2}{n^2} - 2\frac{\alpha(n-\alpha-t)}{n^2} - \frac{(n-\alpha-t)^2}{4n^2} + 4\frac{\alpha}{n} + \frac{5(n-\alpha-t)}{3n} - 1 < 0 \tag{6.17}$$

as well as

$$1 - \frac{3(n-\alpha-t)}{2n} - 2\frac{\alpha}{n} \geq 0. \tag{6.18}$$

In order to compare it to our result, we transform the above inequalities into conditions on t: Then the first inequality (6.17) becomes

$$3\alpha - \frac{7n}{3} - \frac{4\sqrt{-6\alpha n + 4n^2}}{3} < t < 3\alpha - \frac{7n}{3} + \frac{4\sqrt{-6\alpha n + 4n^2}}{3}, \tag{6.19}$$

inequality (6.18) can be transformed into

$$\frac{1}{3}n + \frac{1}{3}\alpha \leq t. \tag{6.20}$$

[1]http://magma.maths.usyd.edu.au/magma/

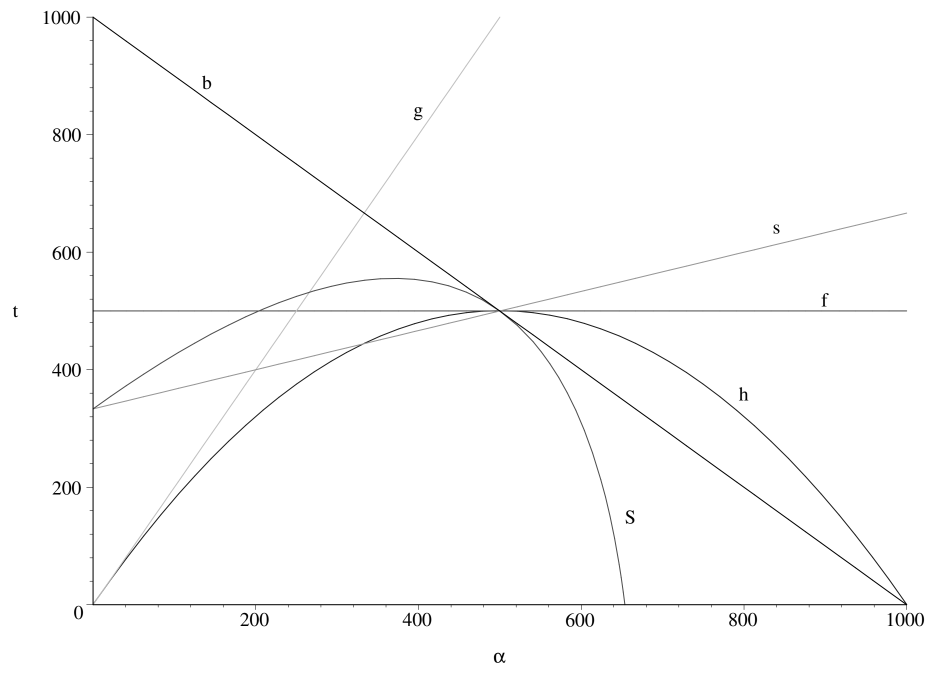

Figure 6.3: $b : t < n - \alpha$, $f : t > \frac{n}{2}$, $g : t > 2\alpha$, $h : t > 2\alpha\left(1 - \frac{\alpha}{n}\right) + \frac{3}{2}$
Upper bound S and lower bound s on t given in equations (6.19) and (6.20), compare [SM09].

Comparison of the results of [SM09] with our modular and integer ones for $k = 1$ with $n = 1000$.

A comparison of these results to ours is given in Table 6.1. We give lower bounds on t for some values of α. The values of α which are considered were also analyzed in the work of Santanu Sarkar and Subhamoy Maitra.

A general comparison between the different conditions on t can be seen in Figure 6.3.

It is

$$\begin{aligned} \frac{1}{3}n + \frac{1}{3}\alpha \quad &< \quad \frac{4\alpha n + 3n - 4\alpha^2}{2n} \\ \Leftrightarrow \quad \frac{5n - \sqrt{n^2 + 108n}}{12} \quad &< \alpha < \quad \frac{5n + \sqrt{n^2 + 108n}}{12} \,. \end{aligned}$$

This implies that our bound is better in case of $\alpha \leq \frac{5n - \sqrt{n^2+108n}}{12}$. This includes most of the possible values of α. The bound obtained in [SM09] is better in the remaining cases. However, the advantage is only slight. Moreover, this is exactly the region in which the experiments in [SM09] do not correspond to their theoretical result but give worse bounds.

6.1.2 The Case of More than One Equation

Having analyzed the problem of implicit factoring with one oracle call, we extend the analysis to more oracle calls. In the modular case the bounds we have obtained improved the more oracle calls were made. We hope to obtain a similar result here as more information is given by more oracle calls. As in the previous section we start with the equation $f_1(x_0, x_1, y_1) := 2^{t_p} x_0 x_1 y_1 + N_0 x_1 - N_1 x_0 = 0$, which we derived from the two moduli N_0 and N_1. By construction of f_1, we know that $f_1(q_0, q_1, \tilde{p}_1 - \tilde{p}_0) = 0$. By any further oracle call we obtain another equation $f_i(x_0, x_i, y_i) := 2^{t_p} x_0 x_i y_i + N_0 x_i - N_i x_0 = 0$ with solutions $(q_0, q_i, \tilde{p}_i - \tilde{p}_0)$, $i = 2, \ldots, k$, where k is the number of oracle calls.

Our goal is as follows. Given the polynomial equations $f_1(x_0, x_1, y_1) := 2^{t_p} x_0 x_1 y_1 + N_0 x_1 - N_1 x_0 = 0, \ldots, f_k(x_0, x_k, y_k) := 2^{t_p} x_0 x_k y_k + N_0 x_k - N_k x_0 = 0$, determine conditions such that we can find $(q_0, q_1, \ldots, q_k, \tilde{p}_1 - \tilde{p}_0, \ldots, \tilde{p}_k - \tilde{p}_0)$ efficiently by Coppersmith's method. Let again $X_i = 2^\alpha$ and $Y_i = 2^{n-\alpha-t}$ denote the upper bounds on the absolute values of the solutions q_i and $\tilde{p}_i - \tilde{p}_0$, respectively. Like in the case of one equation we introduce the following heuristic.

Assumption 6.1.6
Let $\mathcal{F}$ denote a shift polynomial set which is constructed with respect to some polynomials $f_1, \ldots, f_k \in \mathbb{Z}[x_0, \ldots, x_k, y_1, \ldots, y_k]$. Assume that the set $\{f_1, \ldots, f_k, g_1, \ldots, g_\lambda\} \subset \mathbb{Z}[x_0, \ldots, x_k, y_1, \ldots, y_k]$ is the set of polynomials we get by applying Coppersmith's method with the shift polynomial set $\mathcal{F}$. Let $GB := \{h_1, \ldots, h_\kappa\}$ be a Groebner basis of $\{f_1, \ldots, f_k, g_1, \ldots, g_\lambda\}$. Then the factors q_0, $q_1, \ldots, q_k$ can be efficiently determined from GB.

Using this heuristic and the simplest shift polynomial set possible, we can again recalculate the bound $\frac{k+1}{k}\alpha < t$ by applying Coppersmith's method. This is the bound we have obtained by the modular analysis (Theorem 4.2.3).

Theorem 6.1.7
Assume $k \in \mathbb{N}$ to be fixed. As before let $f_i(x_0, x_i, y_i) := 2^{t_p} x_0 x_i y_i + N_0 x_i - N_i x_0$ with $i = 1, \ldots, k$. Furthermore, suppose $\mathcal{F} := \{f_i(x_0, x_i, y_i) \mid i \in \{1, \ldots, k\}\}$. Then, by Coppersmith's method with shift polynomial set $\mathcal{F}$, all common roots $(x_0, x_1, \ldots, x_k, y_1, \ldots, y_k) = (q_0, q_1, \ldots, q_k, \tilde{p}_1 - \tilde{p}_0, \ldots, \tilde{p}_k - \tilde{p}_0)$ with $|q_i| \leq 2^\alpha$ and $|\tilde{p}_i - \tilde{p}_0| \leq 2^{n-\alpha-t}$ can be determined efficiently if

$$\frac{k+1}{k}\alpha + \delta_k < t$$

for a small integer δ_k and if Assumption 6.1.6 holds.

Proof: The proof is analogous to the proof in the case of only one equation $f_1(x_0, x_1, y_1) = 0$. The only difference is that we extend our lattice basis. That is, we add the information we get by the further equations $f_i(x_0, x_i, y_i) = 0$, $i = 2, \ldots, k$. Thus, we set

$$\mathbf{B} := \begin{pmatrix} (X_0X_1Y_1)^{-1} & 0 & 0 & 2^{t_p} & & 0 \\ 0 & \ddots & 0 & 0 & \ddots & 0 \\ 0 & & (X_0X_kY_k)^{-1} & 0 & 0 & 2^{t_p} \\ 0 & & 0 & (X_0)^{-1} & -N_1 \; \ldots & -N_k \\ 0 & & 0 & 0 & N_0 & 0 \\ 0 & & 0 & 0 & 0 \; \ddots & 0 \\ 0 & & 0 & 0 & 0 & N_0 \end{pmatrix} \begin{array}{c} x_0x_1y_1 \\ \vdots \\ x_0x_ky_k \\ x_0 \\ x_1 \\ \vdots \\ x_k \end{array} .$$

The matrix $\mathbf{B}$ is a basis matrix of a lattice L containing the vector

$$\begin{aligned} \mathbf{t} &= (q_0q_1(\tilde{p}_1 - \tilde{p}_0), \ldots, q_0q_k(\tilde{p}_k - \tilde{p}_0), q_0, q_1, \ldots, q_k)\mathbf{B} \\ &= (\frac{q_0q_1(\tilde{p}_1 - \tilde{p}_0)}{X_0X_1Y_1}, \ldots, \frac{q_0q_k(\tilde{p}_k - \tilde{p}_0)}{X_0X_kY_k}, \frac{q_0}{X_0}, 0, \ldots, 0) \,, \end{aligned}$$

which is our target vector. The vector $\mathbf{t}$ is part of a sublattice L_S of L such that the last k entries in vectors of L_S are zero. Let $\mathbf{B_S}$ be a basis matrix of the lattice L_S. According to Lemma 2.3.10, we get a new equation $f_n(x_0, \ldots, x_k, y_1, \ldots, y_k) = 0$ with the same solutions if

$$||\mathbf{t}|| < \det(L_S)^{\frac{1}{k+1}} 2^{-\frac{k}{4}} \,. \tag{6.21}$$

The determinant of the sublattice is

$$\det(L_S) = \det(L) = \det(\mathbf{B}) = X_0^{-(k+1)} \prod_{i=1}^{k} \left(X_i^{-1} Y_i^{-1}\right) N_0^k \,.$$

Therefore, using $N_i \geq 2^{n-1}$, condition (6.21) becomes

$$\begin{aligned} (k+1)^{\frac{1}{2}} &< 2^{\frac{1}{k+1}(-(2k+1)\alpha - k(n-\alpha-t) + k(n-1)) - \frac{k}{4}} \\ \Leftrightarrow \quad \frac{k+1}{k}\alpha + \frac{(k+1)\log{(k+1)}}{2k} + \frac{k+1}{4} + 1 &< t \,, \end{aligned}$$

As α and t are integers as they describe numbers of bits, we require

$$\frac{k+1}{k}\alpha + \delta_k < t \,,$$

where δ_k is the first integer larger than $\frac{(k+1)\log(k+1)}{2k} + \frac{k+1}{4} + 1$.
The computation complexity is dominated by the complexity of the LLL-reduction. As the lattice under consideration is of constant dimension, the running time is polynomial in the bitsize of the values in $\mathbf{B}$. That is, it is polynomial in n. Let GB denote a Groebner basis of $\{f_n, f_1, \ldots, f_k\}$. It can be computed efficiently for fixed values of k as we are dealing with polynomials of maximum degree 3 in $2k+1$ variables. If Assumption 6.1.6 holds, then we can efficiently determine the values $q_0, q_1, \ldots, q_k$ from GB and the claim follows. ■

The above result is rather obvious. When we analyzed the problem of implicit factoring with a modular approach in Section 4.2, the equations were linearized in order to use them in the description of a lattice. Here, using Coppersmith's method with $\{f_1, \ldots, f_k\}$ as shift polynomial set, we do essentially the same. Instead of eliminating the non-linear monomial by the modular operation, we use it as a separate element. As we do not take its structure and relation to other monomials into account, we treat it like an additional variable. That is, we implicitly linearize the given problem. Thus, we obtain essentially the same bound as before.

bitsize α of the q_i	number of moduli $k+1$	bound $\frac{k+1}{k}\alpha$	number of shared bits t	success rate
250	3	375	377	33%
250	3	375	378	75%
350	10	389	390	0%
350	10	389	391	100%
400	100	405	409	0%
400	100	405	410	95%
440	50	449	452	0%
440	50	449	453	80%
480	100	485	491	33%
480	100	485	492	90%

Table 6.2: Attack for imbalanced RSA moduli using Coppersmith's method

Again, we have verified the heuristic given in Assumption 6.1.6 in practice by running experiments on a Core2 Duo 1.66GHz notebook. The attacks were implemented using Magma[2] Version 2.11 using the implementation of L^2 by Nguyen and Stehlé [NS05]. We performed experiments for 1000-bit moduli N_i with various bitsizes of the q_i. We have tested for the same parameter sets as in Section 4.2. The results are presented in Table 6.2.

In the previous section we have seen that we could easily improve the first bound by using special shift polynomials in Coppersmith's method. We would like to do this with respect to systems of equations as well. However, we have to choose the shift polynomial sets carefully. Otherwise we do not have a determinant preserving set and, therefore, cannot calculate the correct bounds. Then the real bounds are worse than predicted. Sufficient criteria of being determinant preserving have been described in Section 5.1.

As we do not have any general strategy to apply Coppersmith's method in case of more than one equation, we now restrict to the case of $k = 2$. By this, we can better see what happens if more equations are used. Further, we can observe how the criteria on systems of equations we determined in Section 5 influence the analysis.

[2]http://magma.maths.usyd.edu.au/magma/

As a first approach, we just shift both polynomials separately with the same shifts we have used to obtain the bound of Theorem 6.1.5. This is possible as a set of this kind remains determinant preserving. Each new shift gives new monomials. The only monomials occurring in shifts of f_1 as well as in shifts of f_i, $i = 2, \dots, k$, are powers of x_0. The number of these monomials, however, does not influence the bound asymptotically. The bounds we get are essentially the same we had in the case of only one equation as the following lemma shows. The only difference is a small additional constant.

Lemma 6.1.8
Let $k = 2$ and suppose Assumption 6.1.6 holds. Let $N_0, N_1, N_2 \in \mathbb{N}$ be of size 2^n. Furthermore, for $\iota = 1, 2$ let $f_\iota(x_0, x_\iota, y_\iota) = 2^{t_p} x_0 x_\iota y_\iota + N_0 x_\iota - N_\iota x_0 \in \mathbb{Z}[x_0, x_1, x_2, y_1, y_2]$. Assume $X_i = 2^\alpha$, $i = 0, 1, 2$, $Y_i = 2^{n-t-\alpha}$, $i = 1, 2$, are bounds on the absolute values of the solutions q_0, q_1 and q_2, $\tilde{p}_1 - \tilde{p}_0$ and $\tilde{p}_2 - \tilde{p}_0$, respectively. Then for all $\epsilon > 0$ there exists $l(\epsilon)$ such that we can determine all integer solutions $(q_0, q_1, q_2, \tilde{p}_1 - \tilde{p}_0, \tilde{p}_2 - \tilde{p}_0)$ such that $|q_i| \le X_i$ and $|\tilde{p}_i - \tilde{p}_0| \le Y_i$ in time polynomial in $n, l(\epsilon)$ if

$$2\alpha \left(1 - \frac{\alpha}{n}\right) + 2 < t - \epsilon\,.$$

Proof: The proof proceeds analogously to the proof of Theorem 6.1.5. We consider the polynomials f_1 and f_2 separately and take optimized shift sets for each of the polynomials. For a better understanding we repeat some important steps of the proof of Theorem 6.1.5 and point out the differences which occur due to the use of two polynomials. Let $l \in \mathbb{N}, \tau \in \mathbb{R}$ such that $\tau l \in \mathbb{N}$. We again construct the lattice by shifting f_ι, $\iota := 1, 2$, with all monomials in the set $S_{l,\tau l}(f_\iota) = \{x_0^i x_\iota^j y_\iota^k \mid k = 0, \dots, l + 2\tau l; i = \max\{0, k - 2\tau l\}, \dots, \min\{l + \tau l, l + k\}$ and $j = l + k - i\}$. Let $n_l(f_\iota) := |S_{l,\tau l}(f_\iota)|$ be the number of monomials used as shifts of f_ι. Then the total number of shift polynomials is

$$n_l := n_l(f_1) + n_l(f_2) = 2\left(\tau^2 l^2 + 2\tau l + \frac{3}{2} l + 1 + \frac{1}{2} l^2 + 2\tau l^2\right) = 2\tau^2 l^2 + 4\tau l + 3l + 2 + l^2 + 4\tau l^2\,.$$

We enumerate the monomials like in the proof of Theorem 6.1.5 and build a lattice basis $\mathbf{B}$ such that we can determine the determinant of the lattice easily. Then we use Coppersmith's algorithm with this lattice basis and, by this, obtain conditions on the sizes of the unknowns.

Here we just present the basic steps of the determinant calculation. This will give us the bound. We reuse results of the proof of Theorem 6.1.5.

The powers of any of $X_0^{-1}, X_1^{-1}, X_2^{-1}, Y_1^{-1}$ and Y_2^{-1} occurring on the diagonal of $\mathbf{B}$ are the powers of the variables occurring in $(S_{l+1,\tau l}(f_1) \setminus M(f_1)) \cup (S_{l+1,\tau l}(f_2) \setminus M(f_2))$. Here, the set $M(f_\iota) := \{x_0^i x_\iota^j y_\iota^k \mid k = 0, \dots, l + 2\tau l; i = \max\{0, k - \tau l\}, \dots, \min\{l + \tau l, k + l\}$ and $j = l + 1 + k - i\}$ denotes the set of all monomials introduced by the shift polynomials with respect to f_ι. However, the two sets $(S_{l+1,\tau l}(f_1) \setminus M(f_1))$ and $(S_{l+1,\tau l}(f_2) \setminus M(f_2))$ are not disjoint. Monomials which occur in both sets may only contain the shared variable x_0. Therefore, the only monomial in both sets is x_0^{l+1}.

Let s_0, s_1 and s_2 denote the powers of X_0^{-1}, X_1^{-1} and X_2^{-1}, respectively, and u_1 and u_2

denote the powers of Y_1^{-1} and Y_2^{-1}. As the polynomials have been regarded separately, the total number of powers of X_1^{-1} is exactly the total number of powers of X_1^{-1} in the proof of Theorem 6.1.5. The value of s_2 equals s_1. The same observations hold for u_1 and u_2. Thus,

$$s_1 = s_2 = 1 + \frac{3}{2}l + \frac{1}{2}l^2 + \frac{3}{2}\tau l + \tau l^2 + \frac{1}{2}\tau^2 l^2\,, \tag{6.22}$$

$$u_1 = u_2 = 1 + \frac{3}{2}l + \frac{1}{2}l^2 + 3\tau l + 2\tau l^2 + 2\tau^2 l^2\,. \tag{6.23}$$

In contrast to the other variables, we have to double the number of powers of X_0^{-1} determined in Theorem 6.1.5 as this variable occurs in both equations. To calculate s_0, we then have to subtract $l+1$, which is the power of X_0^{-1} in the shared monomial. This implies

$$\begin{aligned} s_0 &= 2\left(2 + 3l + l^2 + \frac{7}{2}\tau l + 3\tau l^2 + \frac{3}{2}\tau^2 l^2\right) - (l+1) \\ &= 3 + 5l + 2l^2 + 7\tau l + 6\tau l^2 + 3\tau^2 l^2\,. \end{aligned} \tag{6.24}$$

Using these values and substituting $X_0 = X_1 = X_2 = 2^{\alpha}$, $Y_1 = Y_2 = 2^{n-\alpha-t}$ as well as $N_0 \geq 2^{n-1}$, we obtain

$$\det(L) \geq 2^{t\left(2+3l+l^2+6\tau l+4\tau l^2+4\tau^2 l^2\right)-\alpha\left(4\tau l^2+4\tau l+2l^2+5l+3\right)-n\left(2\tau^2 l^2+2\tau l\right)-\left(2\tau^2 l^2+4\tau l+3l+2+l^2+4\tau l^2\right)}\,.$$

The dimension of the lattice in this case is $r_l = |S_{l+1,\tau l}(f_1) \cup S_{l+1,\tau l}(f_2)| = |S_{l+1,\tau l}(f_1)| + |S_{l+1,\tau l}(f_2)| - 1 = 2\left(4\tau l + 3 + \tau^2 l^2 + 2\tau l^2 + \frac{1}{2}l^2 + \frac{5}{2}l\right) - 1 = 8\tau l + 5 + 2\tau^2 l^2 + 4\tau l^2 + l^2 + 5l$. Consequently, the size of the sublattice is

$$\begin{aligned} r &= r_l - n_l \\ &= \left(8\tau l + 5 + 2\tau^2 l^2 + 4\tau l^2 + l^2 + 5l\right) - \left(2\tau^2 l^2 + 4\tau l + 3l + 2 + l^2 + 4\tau l^2\right) \\ &= 4\tau l + 2l + 3\,. \end{aligned}$$

All components of the target vector $\mathbf{t}$ are smaller than 1 so that $||\mathbf{t}|| \leq \sqrt{4\tau l + 2l + 3}$. Therefore, we get the following condition on the existence of a third equation:

$$\sqrt{4\tau l + 2l + 3} < 2^{-\frac{4\tau l + 2l + 2}{4}} \cdot \left(2^{t\left(2+3l+l^2+6\tau l+4\tau l^2+4\tau^2 l^2\right)-\alpha\left(4\tau l^2+4\tau l+2l^2+5l+3\right)-n\left(2\tau^2 l^2+2\tau l\right)-\left(2\tau^2 l^2+4\tau l+3l+2+l^2+4\tau l^2\right)}\right)^{\frac{1}{4\tau l+2l+3}}$$

$$\begin{aligned} \Leftrightarrow \quad & \frac{4\tau l + 2l + 3}{2}\log\left(4\tau l + 2l + 3\right) + \frac{(4\tau l + 2l + 3)(4\tau l + 2l + 2)}{4} \\ & + \left(2\tau^2 l^2 + 4\tau l + 3l + 2 + l^2 + 4\tau l^2\right) \\ < \quad & t\left(2\tau l + l + 2\right)\left(2\tau l + l + 1\right) - \alpha\left(4\tau l + 2l + 3\right)\left(l + 1\right) - n\left(2\tau^2 l^2 + 2\tau l\right)\,. \end{aligned}$$

As
$$\frac{\frac{(4\tau l+2l+3)(4\tau l+2l+2)}{4}+\left(2\tau^2l^2+4\tau l+3l+2+l^2+4\tau l^2\right)}{(2\tau l+l+2)\,(2\tau l+l+1)}\le 1+1=2\,,$$
we use the slightly stronger constraint
$$\underbrace{\frac{(4\tau l+2l+3)\,(l+1)}{(2\tau l+l+2)\,(2\tau l+l+1)}\alpha+\frac{\left(2\tau^2l^2+2\tau l\right)}{(2\tau l+l+2)\,(2\tau l+l+1)}n+2+\frac{\frac{4\tau l+2l+3}{2}\log\left(4\tau l+2l+3\right)}{(2\tau l+l+2)\,(2\tau l+l+1)}}_{=:h(\alpha,n,l,\tau)}<t\,.$$

This implies that for fixed l and τ we can factor the N_i if t is greater than $h(\alpha,n,l,\tau)$. It is
$$\bar{h}(\alpha,n,\tau):=\lim_{l\to\infty}h(\alpha,n,l,\tau)=\frac{8\tau^2+8\tau+2+4\tau\alpha+2\alpha+2\tau^2n}{4\tau^2+4\tau+1}\,.$$
Let $\epsilon>0$. For $l>l(\epsilon)$ the condition $h(\alpha,n,l,\tau)<t$ simplifies to
$$\frac{8\tau^2+8\tau+2+4\tau\alpha+2\alpha+2\tau^2n}{4\tau^2+4\tau+1}<t-\epsilon\,. \tag{6.25}$$

The optimal value τ_{opt} of τ, such that the lower bound on the number of shared bits $\bar{h}(\alpha,n,\tau_{\text{opt}})$ is as small as possible, is $\tau_{\text{opt}}=\frac{\alpha}{n-2\alpha}$ for $\alpha\neq\frac{n}{2}$. Using this value in the function, we can determine the lower bound on t as
$$\bar{h}(\alpha,n,\tau_{opt})=\frac{2n+2\alpha n-2\alpha^2}{n}<t-\epsilon\,.$$
This is equivalent to
$$\bar{h}(\alpha,n,\tau_{opt})=2\alpha\left(1-\frac{\alpha}{n}\right)+2<t-\epsilon\,. \tag{6.26}$$
On the heuristic Assumption 6.1.6, this implies that we are able to determine the solution and proves the lemma. The running time is dominated by the running time of the LLL-reduction, which is polynomial in n and $l(\epsilon)$. ■

Defining the shift polynomial sets of f_i and f_j, $i\neq j$, independently, we have seen that it does not make a difference if we take one or two polynomials. The bound we obtain is asymptotically the same, except for a small constant. The same property holds if we use more equations. That is, a result analogue to that of Theorem 6.1.5 can be obtained with k oracle calls as well. Theorem and proof can be given completely analogously to Lemma 6.1.8 using k instead of 2 equations. However, as this phenomenon could already be seen with two equations we have only presented that proof. For practical analyses, combining equations this way does not help. Therefore, one would only combine two moduli N_i, and analyze the equation derived from these moduli. By combining equations, only the lattice dimension increases without any improvement with respect to the bound.

Consequently, we need to pursue different approaches in order to take advantage of having more than one equation. We again restrict our considerations to two equations to get

a better insight into this approach. Intuitively, we ought to construct shift polynomial sets comprising more common monomials. The key idea is to use x_2 and y_2 in the shift monomials applied to f_1 and x_1 and y_1 in the shift monomials applied to f_2. By this, more monomials occur several times in various of the shift polynomials. These monomials have to occur on the diagonal only once. Then we can reuse them for other shift polynomials. By this, the value of the determinant should increase. Thus, the bound should be improved. The choice of such a set of shift polynomials, however, has to be made extremely carefully in order to keep the set determinant preserving.
Let us consider an at first sight tempting approach. We shift f_1 by x_2y_2 and f_2 by x_1y_1 such that the monomial of highest degree is the same monomial $x_0x_1x_2y_1y_2$ in both new polynomials. Furthermore, the new polynomials already contain the leading monomials $x_0x_1y_1$ of f_1 and $x_0x_2y_2$ of f_2. Thus, setting $\mathcal{F} := \{x_2y_2f_1, x_1y_1f_2, f_1, f_2\}$ we hope to obtain a good bound.

Example 6.1.9
Let $\mathcal{F} := \{x_2y_2f_1(x_0, x_1, y_1), x_1y_1f_2(x_0, x_2, y_2), f_2(x_0, x_2, y_2), f_1(x_0, x_1, y_1)\}$ denote a set of shift polynomials. Let us enumerate the monomials which occur in $\mathcal{F}$ as $x_0x_1x_2y_1y_2$, $x_0x_2y_2$, $x_0x_1y_1$, x_0, $x_1x_2y_2$, $x_1x_2y_1$, x_2, x_1. Let $\mathbf{B}$ denote a lattice basis constructed to apply Coppersmith's method. In the construction, the column vectors $\mathbf{f}^T$ to build $\mathbf{F}$ are defined with respect to the above enumeration of monomials. Then the value N_0 occurs four times on the diagonal of $\mathbf{F}$. The product of the diagonal elements of $\mathbf{D}$ is $(X_0^4X_1^2X_2^2Y_1^2Y_2^2)^{-1}$. Applying the simplified condition $\det(B) > 1$, this leads to the bound $\alpha < t$. This result is achieved only with infinitely many moduli using the modular analysis (Theorem 4.2.5). Unfortunately, the bound is not correct as the set $\mathcal{F}$ is not determinant preserving: We have that $2^{t_p}x_1y_1f_2 - 2^{t_p}x_2y_2f_1 + N_2f_1 - N_1f_2 \equiv 0 \pmod{N_0}$. Thus, the determinant of the sublattice differs by a factor of N_0 from the one we expected.
If we take only three of the polynomials, e. g. if we define the shift polynomial set $\mathcal{F}' := \{x_2y_2f_1(x_0, x_1, y_1), f_1(x_0, x_1, y_1), f_2(x_0, x_2, y_2)\}$, then we obtain the following bound:

$$\alpha + \frac{n}{4} < t\,.$$

Here we again used the simplified condition. This bound does not change if we choose another subset of $\mathcal{F}$ with cardinality 3.

Let us compare the bound $\alpha + \frac{n}{4} < t$ to the one obtained in the modular case as well as in Theorem 6.1.7 with $k = 2$. On the one hand, $\alpha + \frac{n}{4} \leq \frac{3}{2}\alpha \Leftrightarrow \frac{n}{2} \leq \alpha$. That is, the new lower bound on t is smaller for α having more than half the bitsize of n. On the other hand, α denotes the size of the smaller factor. Moreover, due to $t \leq n - \alpha$, we have the condition $\alpha + \frac{n}{4} \leq n - \alpha \Leftrightarrow \alpha \leq \frac{3}{8}n$. Consequently, the bound given in Example 6.1.9 is worse than the bound $\frac{3}{2}\alpha < t$ for any possible value of α and large enough values of n. We have to search for other, more useful sets of shift polynomials instead.
Let us consider a different example.

$$\begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -N_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -N_2 & 0 & 0 & 0 & 0 & 0 & 2^{t_p} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2^{t_p} & 0 & 0 & 2^{t_p} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -N_1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -N_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2^{t_p} & -N_1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -N_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2^{t_p} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -N_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -N_2 & 0 & 0 & 0 & 0 & 0 & 0 & 2^{t_p} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 2^{t_p} & 2^{t_p} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2^{t_p} & 0 & 0 & -N_2 & 0 & 0 & 0 & 0 & 0 \\
-N_1 & 0 & 0 & 0 & 2^{t_p} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
N_0 & 0 & 0 & 0 & 0 & 0 & 2^{t_p} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & N_0 & 0 & 0 & 0 & 2^{t_p} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & N_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & N_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & N_0 & -N_2 & -N_1 & N_0 & 0 & 0 & 0 & 0 & 0 & 0 & 2^{t_p} & 2^{t_p} & 0 \\
0 & 0 & 0 & 0 & 0 & N_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & N_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -N_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & N_0 & 0 & 0 & 0 & 0 & -N_2 & 0 & -N_1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & N_0 & 0 & 0 & -N_1 & 0 & -N_2 & 0 & 2^{t_p} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & N_0 & 0 & 2^{t_p} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & N_0 & 0 & 2^{t_p} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & N_0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & N_0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & N_0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & N_0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & N_0
\end{pmatrix}$$

Figure 6.4: *The matrix* $\mathbf{F}$ *induced by the shift polynomial set* $\mathcal{F}$ *given in Example 6.1.10.*

Example 6.1.10
We define our shift polynomial set as

$$\mathcal{F} := \{x_0x_1x_2y_1^2y_2f_1, x_0x_1x_2y_1y_2^2f_2, x_0x_2^2y_2^3f_1, x_0x_1^2y_1^3f_2, x_0x_2y_1y_2f_1, x_1x_2y_1y_2f_2, x_1x_2y_1y_2f_1,$$
$$x_0x_1y_1y_2f_2, x_0y_2f_2, x_0y_1f_1, x_0x_1y_1^2f_1, x_0x_2y_2^2f_2, x_1y_1f_1, x_2y_2f_2, x_1y_1f_2, x_2y_2f_1, f_1\}\,.$$

The bound corresponding to these shift polynomials if we apply the simplified condition is

$$\frac{3}{7}\alpha + \frac{11}{28}n \;<\; t\,.$$

This bound is valid if

$$\frac{3}{7}\alpha + \frac{11}{28}n \leq n - \alpha \;\;\Leftrightarrow\;\; \alpha \leq \frac{17}{40}n\,.$$

In comparison to the original bound from the modular analysis $\frac{3}{2}\alpha < t$*, the new bound is smaller whenever* $\frac{3}{7}\alpha + \frac{11}{28}n < \frac{3}{2}\alpha \Leftrightarrow \frac{11}{30}n \leq \alpha$*. Thus, the new bound improves the old one if* $\frac{11}{30}n < \alpha < \frac{17}{40}n$*.*
Compared to the asymptotic bound, however, this bound is still worse, i. e. $2\alpha\left(1 - \frac{\alpha}{n}\right) + \frac{3}{2} < \frac{3}{7}\alpha + \frac{11}{28}n$ *for all positive values of* α*.*

This approach is rather arbitrary. To construct the shift polynomial set, we have started by shifting both polynomials by monomials of the form $(x_0y_1)^{i_1}(x_0y_2)^{i_2}(x_1y_1)^{j_1}(x_2y_2)^{j_2}$. We fixed 6 as an upper bound on the maximum total degree of the shift monomials. Additionally, we excluded all shifts that either introduced too many new monomials or were linearly dependent of the others. Therefore, $\mathcal{F}$ is determinant preserving. Moreover, we know from experiments that the attack works. However, there is no obvious reason why to choose exactly these shifts. Exchanging some of the polynomials in the shift polynomial set may even lead to the same bound. Furthermore, there is no simple generalization to more shift polynomials. However, we need a way to generalize the method as with fixed lattice dimension we cannot beat the asymptotic bound obtained with one equation. In any generalization, however, we have to cope with one major problem: Whenever the degree of the shift monomials increases, we have to take care that we do not include shifts which are linearly dependent of the others modulo some coefficient.

As we do not see a good generalization of the previous approach, we pursue a different one. We aim at reducing the number of variables in order to simplify the problem. In this process, we have to ensure that the polynomials still share some variables. If not, a combined analysis does not improve the bound compared to separate analyses. We set

$$\begin{aligned} g_1(u_1,u_2,z_1,z_2) &:= y_2f_1(x_0,x_1,y_1) = 2^{t_p}\underbrace{x_0y_1}_{=:u_1}\underbrace{x_1y_2}_{=:z_1}+N_0\underbrace{x_1y_2}_{=:z_1}-N_1\underbrace{x_0y_2}_{=:u_2}, \\ g_2(u_1,u_2,z_1,z_2) &:= y_1f_2(x_0,x_2,y_2) = 2^{t_p}\underbrace{x_0y_2}_{=:u_2}\underbrace{x_2y_1}_{=:z_2}+N_0\underbrace{x_2y_1}_{=:z_2}-N_2\underbrace{x_0y_1}_{=:u_1}. \end{aligned} \tag{6.27}$$

Common roots of the polynomials g_1 and g_2 are given by $\bar{u}_1 = q_0(\tilde{p}_1-\tilde{p}_0)$, $\bar{u}_2 = q_0(\tilde{p}_2-\tilde{p}_0)$, $\bar{z}_1 = q_1(\tilde{p}_2-\tilde{p}_0)$ and $\bar{z}_2 = q_2(\tilde{p}_1-\tilde{p}_0)$. Hence, the upper bounds on the solutions we are searching for are given by $|\bar{u}_i| \leq 2^{\alpha+n-\alpha-t} = 2^{n-t} =: U_i$, and, analogously, $|\bar{z}_i| \leq 2^{n-t} =: Z_i$, $i = 1,2$. Note that these bounds are independent of α. Thus, by Coppersmith's method we are able to determine all solutions fulfilling these bounds if $t > \psi(n)$ for some function $\psi(n)$. Using this technique, we can obtain quite good bounds for small dimensional lattices. For some values of α these bounds are better compared to other bounds obtained so far. However, these explicit bounds do not beat the general asymptotic bound either.

Example 6.1.11
Let the shift polynomial set be defined as $\mathcal{F} := \{u_2z_2g_1, u_1g_1, u_2g_2, z_1g_2, g_1, g_2\}$. *With an*

appropriate enumeration of the monomials in $\mathcal{F}$ we define a basis $\mathbf{B}$ of a lattice L:

$$\left(\begin{array}{ccccccc}
 & 2^{t_p} & & & & & \\
 & & 2^{t_p} & & & & \\
\mathbf{D} & -N_1 & & 2^{t_p} & & & \\
 & & -N_1 & -N_2 & & & \\
 & & & & & & -N_2 \\
 & & & & & -N_1 & \\
 & N_0 & & & 2^{t_p} & & \\
 & & N_0 & & -N_2 & 2^{t_p} & \\
 & & & N_0 & & & 2^{t_p} \\
 & & & & N_0 & & \\
 & & & & & N_0 & \\
 & & & & & & N_0
\end{array}\right)
\begin{array}{c}
u_1u_2z_1z_2 \\ u_1^2z_1 \\ u_2^2z_2 \\ u_1u_2 \\ u_1 \\ u_2 \\ u_2z_1z_2 \\ u_1z_1 \\ u_2z_2 \\ z_1z_2 \\ z_1 \\ z_2
\end{array} \qquad (6.28)$$

Here, $\mathbf{D} := Diag((U_1U_2Z_1Z_2)^{-1}, (U_1^2Z_1)^{-1}, (U_2^2Z_2)^{-1}, (U_1U_2)^{-1}, (U_1)^{-1}, (U_2)^{-1})$. We calculate conditions on the parameters on which we can determine the solution $(\bar{u}_1, \bar{u}_2, \bar{z}_1, \bar{z}_2)$ using the lattice L. First, note that $\mathcal{F}$ is determinant preserving. This can be seen easily when regarding the matrix. As any shift polynomial introduces a new monomial with coefficient N_0, modular dependence only has to be checked modulo N_0: Let $a_1u_2z_2g_1 + a_2u_1g_1 + a_3u_2g_2 + a_4z_1g_2 + a_5g_1 + a_6g_2 \equiv 0 \pmod{N_0}$. Coefficientwise comparison gives $a_62^{t_p} \equiv 0 \pmod{N_0}$ (as coefficient of u_2z_2), $a_5(-N_1) \equiv 0 \pmod{N_0}$ (as coefficient of u_2), $a_42^{t_p} \equiv 0 \pmod{N_0}$ (as coefficient of $u_2z_1z_2$), $a_12^{t_p} \equiv 0 \pmod{N_0}$ (as coefficient of $u_1u_2z_1z_2$), and $a_22^{t_p} \equiv 0 \pmod{N_0}$ (as coefficient of $u_1^2z_1$). Thus, $a_1 \equiv a_2 \equiv a_4 \equiv a_5 \equiv a_6 \equiv 0 \pmod{N_0}$. Using these results and $-N_1a_2 - N_2a_3 \equiv 0 \pmod{N_0}$ (coefficient of u_1u_2), we additionally obtain $a_3 \equiv 0 \pmod{N_0}$. Thus, the polynomials of $\mathcal{F}$ are linearly independent modulo N_0.
Consequently, $\mathcal{F}$ is determinant preserving and we can directly use the determinant of L to calculate conditions on which we can determine the solutions.
It is $\det(L) = U_1^{-5}U_2^{-5}Z_1^{-2}Z_2^{-2}N_0^6 = 2^{-14(n-t)+6n}$. We again use the simplified condition $\det(L) > 1$ and obtain

$$\frac{4}{7}n < t\,. \qquad (6.29)$$

This condition can be fulfilled by parameter sets with $\frac{4}{7}n < n - \alpha$, thus, whenever $\alpha < \frac{3}{7}n$. Now we would like to compare this bound to the ones we have obtained with the lattice constructions given in the previous examples. It is $\frac{4}{7}n \leq \frac{3}{7}\alpha + \frac{11}{28}n \Leftrightarrow \frac{5}{12}n \leq \alpha$ and $\frac{4}{7}n \leq \frac{3}{2}\alpha \Leftrightarrow \frac{8}{21}n \leq \alpha$. Thus, for a small range of larger values of α, the new construction improves the bounds obtained in Theorem 6.1.7 and Example 6.1.10.

Like in the case of Theorem 6.1.7, we have successfully checked the validity of Assumption 6.1.6 with respect to the previous examples. Apart from a few negative examples with a parameter t being tight to the bound, Assumption 6.1.6 could be verified.
A general comparison of the bounds we have obtained in the various examples is given in Figure 6.5. The overall best results obtained non-asymptotically so far are $\frac{3}{2}\alpha < t$ if

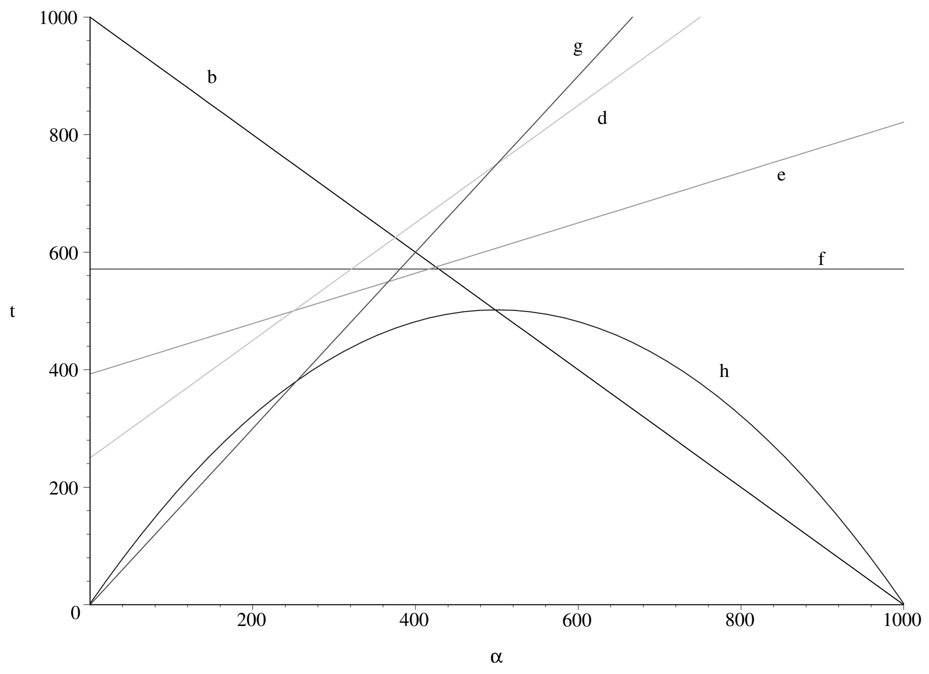

Figure 6.5: $b : t < n - \alpha$, $d : t > \alpha + \frac{n}{4}$, $e : t > \frac{3}{7}\alpha + \frac{11}{28}n$ $f : t > \frac{4}{7}n$, $g : t > \frac{3}{2}\alpha$, $h : t > 2\alpha\left(1 - \frac{\alpha}{n}\right) + 2$

Bounds on the number t of bits to be shared in the larger factor depending on α in the case of two equations and $n = 1000$.

$\alpha < \frac{11}{30}n$, $\frac{3}{7}\alpha + \frac{11}{28}n < t$ if $\frac{11}{30}n \leq \alpha < \frac{5}{12}n$, and $\frac{4}{7}n < t$ if $\frac{5}{12}n \leq \alpha < \frac{3}{7}n$. The first result is represented by function g, the second result by function e, and the third result by function f. Furthermore, the bound calculated in Example 6.1.9 is given by the function d. For comparison, the asymptotic bound $t > 2\alpha\left(1 - \frac{\alpha}{n}\right) + 2$ is also given by function h. We remark that the explicit bounds are quite good in comparison to the bounds obtained with equally low dimensional lattices in case of one equation.

Nevertheless, we are not able to beat the asymptotic bound this way, except for small values of α. In the following paragraphs, we will give an intuition why we cannot obtain a good asymptotic bound with the modified polynomials g_1 and g_2. We use a counting argument to illustrate this.

Let $\mathcal{F}$ denote a shift polynomial set built using g_1 and g_2, and let δ denote the maximum total degree of a monomial occurring in $\mathcal{F}$. As the polynomials g_1 and g_2 have maximum total degree 2, this implies that the maximum total degree of a shift monomial is $\delta - 2$. First, assume that we can take all monomials of degree less or equal to $\delta - 2$ as shift monomials for both polynomials. If $\mathcal{F}$ remained determinant preserving, this would be a sensible strategy as all single variables occur as monomials in either g_1 or g_2.

The number of all monomials $m := u_1^{i_1} u_2^{i_2} z_1^{j_1} z_2^{j_2} \in \mathbb{Z}[u_1, u_2, z_1, z_2]$ with $\deg(m) \leq \delta - 2$ is

$$s := \sum_{i_1=0}^{\delta-2} \sum_{i_2=0}^{\delta-2-i_1} \sum_{j_1=0}^{\delta-2-i_1-i_2} \sum_{j_2=0}^{\delta-2-i_1-i_2-j_1} 1 \ = \ \frac{1}{24}\delta^4 + \frac{1}{12}\delta^3 - \frac{1}{24}\delta^2 - \frac{1}{12}\delta\,.$$

Hence, as we can shift each polynomial by each monomial, the number of elements of $\mathcal{F}$ is bounded by $2s$. Let us regard the polynomials $f \in \mathcal{F}$ as elements of the polynomial ring $\mathbb{Z}_{N_0}[u_1, u_2, z_1, z_2]$. We set $\mathcal{F}_{N_0} := \{f \pmod{N_0} \mid f \in \mathcal{F}\}$. To get a rough estimate how many of these polynomials can be linearly independent modulo N_0, we calculate an upper bound s_M on the number of monomials $|\mathrm{Mon}(\mathcal{F}_{N_0})|$. For any monomial $m := u_1^{i_1} u_2^{i_2} z_1^{j_1} z_2^{j_2} \in \mathrm{Mon}(\mathcal{F}_{N_0})$ one of the following conditions holds:
If $\deg(m) = \delta$, then m can only be obtained by multiplying the leading monomial of either g_1 or g_2 with a monomial of degree $\delta - 2$. Thus, either $i_1 \geq 1$ and $j_1 \geq 1$ or $i_2 \geq 1$ and $j_2 \geq 1$.
If $\deg(m) < \delta$, then either $i_1 \geq 1$ or $i_2 \geq 1$. This is because the monomials of $g_1 \pmod{N_0}$ are $u_1 z_1$ and u_2, and, analogously, the monomials of $g_2 \pmod{N_0}$ are $u_2 z_2$ and u_1. Hence,

$$\begin{aligned} s_M &= \sum_{j_1=1}^{\delta-1} \sum_{i_1=1}^{\delta-j_1} \sum_{i_2=0}^{\delta-i_1-j_1} 1 \quad (j_2 = \delta - i_1 - i_2 - j_1) \\ &+ \sum_{j_1=1}^{\delta-2} \sum_{j_2=1}^{\delta-1-j_1} 1 \quad (i_1 = 0, i_2 = \delta - j_1 - j_2) \\ &+ \sum_{j_2=1}^{\delta-1} \sum_{i_2=1}^{\delta-j_2} 1 \quad (j_1 = 0, i_1 = \delta - i_2 - j_2) \\ &+ \sum_{i_1=1}^{\delta-1} \sum_{i_2=0}^{\delta-1-i_1} \sum_{j_1=0}^{\delta-1-i_1-i_2} \sum_{j_2=0}^{\delta-1-i_1-i_2-j_1} 1 \\ &+ \sum_{i_2=1}^{\delta-1} \sum_{j_1=0}^{\delta-1-i_2} \sum_{j_2=0}^{\delta-1-i_2-j_1} 1 \quad (i_1 = 0) \\ &= \frac{1}{24}\delta^4 + \frac{5}{12}\delta^3 + \frac{23}{24}\delta^2 - \frac{29}{12}\delta + 1\,. \end{aligned}$$

In this description the first three summands correspond to monomials of degree δ, whereas the last two summands correspond to monomials of smaller degree.
For fixed values of δ, a determinant preserving shift polynomial set $\mathcal{F}$ can contain at most s_M polynomials. Any further polynomial f will linearly depend on $\mathcal{F}$ modulo N_0. That is, the additional polynomial f does not give a useful contribution to the determinant of the sublattice. If $\mathcal{F}$ is of size $2s$, then at least

$$2s - s_M \ = \ \frac{1}{24}\delta^4 - \frac{1}{4}\delta^3 - \frac{25}{24}\delta^2 + \frac{9}{4}\delta - 1$$

do not contribute to the determinant due to linear dependence modulo N_0. Note that in case that the value $2s - s_M$ is negative, we cannot exclude any shift polynomials with this argument. However, for all $\delta > 8$, it is $2s - s_M > 0$.
Asymptotically, we obtain $2s - s_M = \frac{1}{24}\delta^4 + o(\delta^4)$. This value is equal to the asymptotic value of s. This implies that the number of shift polynomials we can use is also s, the number of shifts we can maximally construct using only one polynomial. Consequently, even if shifting in both polynomials, we can only use as many shifts of both polynomials as we could construct shifts of one polynomial.
However, regarding a single polynomial, g_i contains as many variables as f_i, namely three. That is, we do no longer have the advantage of having less variables which we had with two polynomials. Moreover, the total sizes of the unknowns are smaller in f_i than in g_i. Thus, better than considering only one polynomial g_i is to use the original polynomial f_i.

Returning to the original polynomials, the question is whether we can argue similarly as in case of the modified polynomials g_i. More precisely, can we give an argument why we cannot improve the bounds? We answer this question in the affirmative. Now let $\mathcal{F}$ denote a shift polynomial set built using f_1 and f_2, and let δ denote the maximum total degree of a monomial occurring in $\mathcal{F}$. As both polynomials have maximum total degree 3, this implies that the maximum total degree of a shift monomial used to construct an element of $\mathcal{F}$ is $\delta - 3$. Let $f \in \mathcal{F}$ be constructed as mf_1 or mf_2, where m is a product of powers of x_0, x_1, x_2, $x_0x_1y_1$, $x_0x_2y_2$, x_0y_1, x_1y_1, x_0y_2 and x_2y_2. That means, the monomial m can be described as $m = x_0^{i_0}x_1^{i_1}x_2^{i_2}y_1^{j_1}y_2^{j_2}$ such that $i_0+i_1+i_2+j_1+j_2 \leq \delta-3$ and $i_0+i_1+i_2 \geq j_1+j_2$. Thus, the maximum number s of all such monomials m is given as

$$\begin{aligned} s &= \sum_{j_2=0}^{\lfloor\frac{\delta-3}{2}\rfloor} \sum_{j_1=0}^{\lfloor\frac{\delta-3}{2}\rfloor-j_2} \sum_{i_2=0}^{j_2} \sum_{i_1=0}^{j_1} \sum_{i_0=j_1-i_1+j_2-i_2}^{\delta-3-i_1-i_2-j_1-j_2} 1 \\ &+ \sum_{j_2=0}^{\lfloor\frac{\delta-3}{2}\rfloor} \sum_{j_1=0}^{\lfloor\frac{\delta-3}{2}\rfloor-j_2} \sum_{i_2=0}^{j_2} \sum_{i_1=j_1+1}^{\delta-3-j_1-2j_2} \sum_{i_0=j_2-i_2}^{\delta-3-i_1-i_2-j_1-j_2} 1 \\ &+ \sum_{j_2=0}^{\lfloor\frac{\delta-3}{2}\rfloor} \sum_{j_1=0}^{\lfloor\frac{\delta-3}{2}\rfloor-j_2} \sum_{i_2=j_2+1}^{\delta-3-j_2-2j_1} \sum_{i_1=0}^{j_1} \sum_{i_0=j_1-i_1}^{\delta-3-i_1-i_2-j_1-j_2} 1 \\ &+ \sum_{j_2=0}^{\lfloor\frac{\delta-3}{2}\rfloor} \sum_{j_1=0}^{\lfloor\frac{\delta-3}{2}\rfloor-j_2} \sum_{i_2=j_2+1}^{\delta-3-j_2-2j_1} \sum_{i_1=j_1+1}^{\delta-3-i_2-j_1-j_2} \sum_{i_0=0}^{\delta-3-i_1-i_2-j_1-j_2} 1 \\ &= \frac{3}{640}\delta^5 - \frac{1}{192}\delta^3 + \frac{1}{1920}\delta\,. \end{aligned}$$

Hence, the number of elements of $\mathcal{F}$ is bounded by $2s$. Again, let us regard them as elements of $\mathbb{Z}_{N_0}[x_0, x_1, x_2, y_1, y_2]$. We set $\mathcal{F}_{N_0} := \{f \pmod{N_0} \mid f \in \mathcal{F}\}$. Then we calculate an upper bound s_M on the number of monomials $|\mathrm{Mon}(\mathcal{F}_{N_0})|$. For any monomial $m := x_0^{i_0}x_1^{i_1}x_2^{i_2}y_1^{j_1}y_2^{j_2} \in \mathrm{Mon}(\mathcal{F}_{N_0})$ one of the following conditions holds:

If $\deg(m) = \delta$, then m is derived from the multiplication of some monomial of degree $\delta - 3$ with the leading monomial of either f_1 or f_2. Thus, either $i_0, i_1 \geq 1$ and $j_1 \geq 1$ or $i_0, i_2 \geq 1$ and $j_2 \geq 1$.
If $\deg(m) < \delta$, then $i_0 \geq 1$. Using this, we calculate the number s_M of monomials which occur in $\mathcal{F}$. The first four sums correspond to monomials of degree smaller than or equal to $\delta - 2$. Monomials of degree $\delta - 1$ do not occur. We get the latter eight sums from monomials of degree δ, the first four corresponding to monomials with i_0, i_1 and $j_1 \geq 1$, the others to the remaining monomials.

$$
\begin{aligned}
s_M = & \sum_{j_2=0}^{\lfloor\frac{\delta-3}{2}\rfloor} \sum_{j_1=0}^{\lfloor\frac{\delta-3}{2}\rfloor-j_2} \sum_{i_2=0}^{j_2-1} \sum_{i_1=0}^{j_1-1} \sum_{i_0=j_1-i_1+j_2-i_2}^{\delta-2-i_1-i_2-j_1-j_2} 1 \\
& + \sum_{j_2=0}^{\lfloor\frac{\delta-3}{2}\rfloor} \sum_{j_1=0}^{\lfloor\frac{\delta-3}{2}\rfloor-j_2} \sum_{i_2=0}^{j_2-1} \sum_{i_1=j_1}^{\delta-2-j_1-2j_2} \sum_{i_0=j_2-i_2}^{\delta-2-i_1-i_2-j_1-j_2} 1 \\
& + \sum_{j_2=0}^{\lfloor\frac{\delta-3}{2}\rfloor} \sum_{j_1=0}^{\lfloor\frac{\delta-3}{2}\rfloor-j_2} \sum_{i_2=j_2}^{\delta-2-j_2-2j_1} \sum_{i_1=0}^{j_1-1} \sum_{i_0=j_1-i_1}^{\delta-2-i_1-i_2-j_1-j_2} 1 \\
& + \sum_{j_2=0}^{\lfloor\frac{\delta-3}{2}\rfloor} \sum_{j_1=0}^{\lfloor\frac{\delta-3}{2}\rfloor-j_2} \sum_{i_2=j_2}^{\delta-3-j_2-2j_1} \sum_{i_1=j_1}^{\delta-3-i_2-j_1-j_2} \sum_{i_0=1}^{\delta-2-i_1-i_2-j_1-j_2} 1 \\
& + \sum_{j_2=0}^{\lfloor\frac{\delta}{2}\rfloor-1} \sum_{j_1=1}^{\lfloor\frac{\delta}{2}\rfloor-j_2} \sum_{i_2=0}^{j_2-1} \sum_{i_1=1}^{j_1-1} 1 \quad (i_0 = \delta - i_1 - i_2 - j_1 - j_2) \\
& + \sum_{j_2=0}^{\lfloor\frac{\delta}{2}\rfloor-1} \sum_{j_1=1}^{\lfloor\frac{\delta}{2}\rfloor-j_2} \sum_{i_2=0}^{j_2-1} \sum_{i_1=j_1}^{\delta-j_1-2j_2} 1 \quad (i_0 = \delta - i_1 - i_2 - j_1 - j_2) \\
& + \sum_{j_2=0}^{\lfloor\frac{\delta}{2}\rfloor-1} \sum_{j_1=1}^{\lfloor\frac{\delta}{2}\rfloor-j_2} \sum_{i_2=j_2}^{\delta-j_2-2j_1} \sum_{i_1=1}^{j_1-1} 1 \quad (i_0 = \delta - i_1 - i_2 - j_1 - j_2) \\
& + \sum_{j_2=0}^{\lfloor\frac{\delta}{2}\rfloor-1} \sum_{j_1=1}^{\lfloor\frac{\delta}{2}\rfloor-j_2} \sum_{i_2=j_2}^{\delta-j_2-2j_1} \sum_{i_1=j_1}^{\delta-1-i_2-j_1-j_2} 1 \quad (i_0 = \delta - i_1 - i_2 - j_1 - j_2) \\
& + \sum_{j_2=1}^{\lfloor\frac{\delta}{2}\rfloor} \sum_{i_2=1}^{j_2-1} \sum_{i_1=0}^{\delta-2j_2} 1 \quad (j_1 = 0, i_0 = \delta - i_1 - i_2 - j_2) \\
& + \sum_{j_2=1}^{\lfloor\frac{\delta}{2}\rfloor} \sum_{i_2=j_2}^{\delta-1-j_2} \sum_{i_1=0}^{\delta-1-i_2-j_2} 1 \quad (j_1 = 0, i_0 = \delta - i_1 - i_2 - j_2)
\end{aligned}
$$

$$\begin{aligned}
&+ \sum_{j_2=1}^{\lfloor\frac{\delta}{2}\rfloor} \sum_{j_1=1}^{\lfloor\frac{\delta}{2}\rfloor - j_2} \sum_{i_2=1}^{j_2-1} 1 \quad (i_1 = 0, i_0 = \delta - i_1 - i_2 - j_1 - j_2) \\
&+ \sum_{j_2=1}^{\lfloor\frac{\delta}{2}\rfloor} \sum_{j_1=1}^{\lfloor\frac{\delta}{2}\rfloor - j_2} \sum_{i_2=j_2}^{\delta-j_2-2j_1} 1 \quad (i_1 = 0, i_0 = \delta - i_1 - i_2 - j_1 - j_2) \\
&= \frac{3}{640}\delta^5 + \frac{7}{192}\delta^4 + \frac{29}{192}\delta^3 - \frac{127}{192}\delta^2 + \frac{601}{1920}\delta + \frac{7}{128}\,.
\end{aligned}$$

For fixed values of δ, the shift polynomial set $\mathcal{F}$ can contain at most s_M polynomials which are linearly independent modulo N_0. Any further polynomial will not give a helpful contribution to the determinant of the sublattice. If $\mathcal{F}$ is of size $2s$, then at least

$$2s - s_M = \frac{3}{640}\delta^5 - \frac{7}{192}\delta^4 - \frac{31}{192}\delta^3 + \frac{127}{192}\delta^2 - \frac{599}{1920}\delta - \frac{7}{128}$$

polynomials are linearly dependent of the others modulo N_0. Asymptotically, we obtain $2s - s_M = \frac{3}{640}\delta^5 + o(\delta^5)$. Like in the previous case, this is equal to the asymptotic value of s. Thus, the number of shift polynomials we can use is also s, the maximum number of shift polynomials we can build using only one polynomial. Consequently, the absolute number of shift polynomials in a determinant preserving shift polynomial set has to be smaller than the maximum number of possible shifts in one polynomial.
We remark that the argumentation works similarly using all possible monomials of degree $\delta - 3$ to construct the shift polynomial set. The same observations can be made with respect to other shift polynomial sets. However, for this type of polynomials we already know a good shift polynomial set to analyze only one equation. Thus, we have used a naturally generalized version of this set to construct $\mathcal{F}$.

These results indicate that we cannot improve the asymptotic bound obtained with only one polynomial by using additional ones. Note, however, that we do not have to use the same shift monomials for both polynomials. A shift polynomial set consisting of s shift polynomials including shifts of f_1 and of f_2 might still give a better bound than a shift polynomial set containing only shifts of f_1. However, a bound usually gets better if less monomials have to be introduced. As less monomials also imply more linear dependences, we do not see a way how to construct such shift polynomial sets based on f_1 and f_2.
A reason for this behavior is probably the structure of the polynomials we examine. Recall that if we regard the polynomials modulo N_0, we deal with polynomials in only two monomials. More precisely, we consider polynomial equations $2^{t_p}x_0x_iy_i - N_ix_0 \equiv 0 \pmod{N_0}$. A solution of this equation is given by $(x_0, x_i, y_i) = (q_0, q_i, \tilde{p}_i - \tilde{p}_0)$. As q_0 divides N_0, we search for solutions of the modular equation $2^{t_p}x_iy_i - N_i \equiv 0 \pmod{p_0}$. Regarding the equations this way, two equations with different indices i only contain independent variables. This observation supports the claim that we cannot use the shared variables in our analyses.
Note that we do not prove that it is impossible to obtain better bounds than $t - \epsilon >$

$2\alpha\left(1-\frac{\alpha}{n}\right)+\frac{3}{2}$, but that we give arguments to support this claim. A rigorous result is a goal in future work.

6.2 Open Problems and Future Work

In 2007, Aurélie Bauer and Antoine Joux presented a way to use Coppersmith's algorithm with several integer polynomials. The shift polynomial set $\mathcal{F}$ they use is determinant preserving by construction. However, the shift polynomials cannot directly be constructed by multiplying the given polynomials by monomials. Instead, it has to be checked if the ideal I induced by these polynomials is prime. If not, a prime ideal has to be constructed by decomposition of I in prime ideals. Then a prime ideal corresponding to the original polynomials has to be chosen. Subsequently, a Groebner basis of this prime ideal has to be computed.

In this chapter we have analyzed how to use Coppersmith's algorithm with a shift polynomial set $\mathcal{F}$ directly constructed from the integer polynomials. This allows for simpler constructions of shift polynomial sets. However, not all shift polynomial sets constructed this way can be used. Instead, further criteria have to be checked. Namely, arbitrary shift polynomial sets are not necessarily determinant preserving. We have given some criteria to guarantee that a shift polynomial set is determinant preserving. These criteria do not directly lead to a general strategy though.

Therefore, in Section 6.1 we have drawn our attention to the problem of implicit factoring and analyzed how to apply Coppersmith's method with regard to these specific polynomials. We have stated some good bounds obtained with small dimensional lattices. However, we could not give a general strategy to calculate a bound on basis of more polynomials. On the contrary, we have given arguments why a better bound cannot be obtained if we use two polynomials instead of one polynomial. It remains as an open problem to prove or disprove the existence of an attack for a wider range of parameters. That is, a goal for future research is either to extend the arguments to a rigorous proof or to show how to achieve a better bound.

Moreover, it would be interesting to give a better explanation of the behavior we have observed with respect to the problem of implicit factoring. Probably the structure of the polynomials plays an important role. Recall that the polynomials consist of only two monomials modulo N_0. Moreover, the integer polynomial equations imply two modular polynomial equations in independent variables. These observations support the claim that we cannot use the shared variables in our analyses.
Therefore, based on the results in the case of the specific example of implicit factoring, it would be interesting to determine criteria whether an additional equation can be used to improve the bounds or not.

On a more general scale, a main target is to develop a generic strategy how to apply

Coppersmith's algorithm to a system of integer polynomial equations. The method should be direct. That is, one should be able to construct a shift polynomial set directly from the initial polynomials. Moreover, the set determined this way ought to be determinant preserving and, thereby, allow for a simple calculation of the bounds.
Approaching this problem from the opposite direction, a major goal is to give impossibility results. Counting arguments like the ones given in the previous section should be elaborated on to give impossibility results. Then they should be generalized to work with arbitrary sets of polynomials.
In the end, given a set of polynomial equations in $\mathbb{Z}[x_1, \ldots, x_l]$, one should directly be able to determine values $X_1, \ldots, X_l$ such that solutions $|\bar{x}_i| \leq X_i$ can be found efficiently, but solutions $|\bar{x}_i| > X_i$ cannot.

Bibliography

[AM09] Divesh Aggarwal and Ueli M. Maurer. Breaking RSA Generically Is Equivalent to Factoring. In Antoine Joux, editor, *EUROCRYPT*, volume 5479 of *Lecture Notes in Computer Science*, pages 36–53. Springer, 2009.

[Bau08] Aurélie Bauer. *Vers une Géneralisation Rigoureuse des Méthodes de Coppersmith pour la Recherche de Petites Racines de Polynômes, Dissertation.* 2008.

[BD99] Dan Boneh and Glenn Durfee. Cryptanalysis of RSA with Private Key d Less than $N^{0.292}$. In Jacques Stern, editor, *EUROCRYPT*, volume 1592 of *Lecture Notes in Computer Science*, pages 1–11. Springer, 1999.

[Ber67] Elwyn R. Berlekamp. Factoring Polynomials Over Finite Fields. *Bell System Tech. J.*, 46:1849–1853, 1967.

[Ber70] Elwyn R. Berlekamp. Factoring Polynomials Over Large Finite Fields. *Math. of Computation*, 24:713–735, 1970.

[BJ07] Aurélie Bauer and Antoine Joux. Toward a Rigorous Variation of Coppersmith's Algorithm on Three Variables. In Moni Naor, editor, *EUROCRYPT*, volume 4515 of *Lecture Notes in Computer Science*, pages 361–378. Springer, 2007.

[Blö00] Johannes Blömer. Closest Vectors, Successive Minima, and Dual HKZ-Bases of Lattices. In Ugo Montanari, José D. P. Rolim, and Emo Welzl, editors, *ICALP*, volume 1853 of *Lecture Notes in Computer Science*, pages 248–259. Springer, 2000.

[BM05] Johannes Blömer and Alexander May. A Tool Kit for Finding Small Roots of Bivariate Polynomials over the Integers. In Cramer [Cra05], pages 251–267.

[Bon99] Dan Boneh. Twenty Years of Attacks on the RSA Cryptosystem. In *Notices of the American Mathematical Society (AMS)*, volume 46, No. 2, pages 203–213, 1999.

[Bro06] Daniel R. L. Brown. Breaking RSA May Be as Difficult as Factoring. In *Cryptology ePrint Archive, Report 2005/380*, 2006.

[BS96] Eric Bach and Jeffrey Shallit. *Algorithmic Number Theory*, volume 1. The MIT press, 1996.

[Buc65] Bruno Buchberger. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal (An Algorithm for Finding the Basis Elements in the Residue Class Ring Modulo a Zero Dimensional Polynomial Ideal)*. PhD thesis, Mathematical Institute, University of Innsbruck, Austria, 1965.

[Buc06] Bruno Buchberger. Bruno Buchberger's PhD thesis 1965: An Algorithm for Finding the Basis Elements of the Residue Class Ring of a Zero Dimensional Polynomial Ideal. *J. Symb. Comput.*, 41(3-4):475–511, 2006.

[BV98] Dan Boneh and Ramarathnam Venkatesan. Breaking RSA May Not Be Equivalent to Factoring. In Nyberg [Nyb98], pages 59–71.

[BvHKS07] Karim Belabas, Mark van Hoeij, Jürgen Klüners, and Allan Steel. Factoring Polynomials over Global fields. *preprint, to appear in Journal de Theorie des Nombres de Bordeaux*, 2007.

[CFPR96] Don Coppersmith, Matthew K. Franklin, Jacques Patarin, and Michael K. Reiter. Low-Exponent RSA with Related Messages. In Maurer [Mau96], pages 1–9.

[CKM97] Stéphane Collart, Michael Kalkbrener, and Daniel Mall. Converting Bases with the Gröbner Walk. *J. Symb. Comput.*, 24(3/4):465–469, 1997.

[CLO97] David Cox, John Little, and Donald O'Shea. *Ideals, Varieties and Algorithms, An Introduction to computational Algebraic Geometry and Commutative Algebra, Second Edition*. Springer-Verlag, 1997.

[Cop96a] Don Coppersmith. Finding a Small Root of a Bivariate Integer Equation; Factoring with High Bits Known. In Maurer [Mau96], pages 178–189.

[Cop96b] Don Coppersmith. Finding a Small Root of a Univariate Modular Equation. In Maurer [Mau96], pages 155–165.

[Cop97] Don Coppersmith. Small Solutions to Polynomial Equations, and Low Exponent RSA Vulnerabilities. *J. Cryptology*, 10(4):233–260, 1997.

[Cop01] Don Coppersmith. Finding Small Solutions to Small Degree Polynomials. In Silverman [Sil01], pages 20–31.

[Cor04] Jean-Sébastien Coron. Finding Small Roots of Bivariate Integer Polynomial Equations Revisited. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 492–505. Springer, 2004.

[Cor07] Jean-Sébastien Coron. Finding Small Roots of Bivariate Integer Polynomial Equations: A Direct Approach. In Menezes [Men07], pages 379–394.

[Cra05] Ronald Cramer, editor. *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, volume 3494 of *Lecture Notes in Computer Science*. Springer, 2005.

[DBL83] *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing, 25-27 April 1983, Boston, Massachusetts, USA*. ACM, 1983.

[DH76] Whitfield Diffie and Martin E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[DK02] Ivan Damgård and Maciej Koprowski. Generic Lower Bounds for Root Extraction and Signature Schemes in General Groups. In Lars R. Knudsen, editor, *EUROCRYPT*, volume 2332 of *Lecture Notes in Computer Science*, pages 256–271. Springer, 2002.

[Fau99] Jean-Charles Faugère. A New Efficient Algorithm for Computing Gröbner Bases (F4). *Journal of Pure and Applied Algebra*, 139(1–3):61–88, 1999.

[Fau02] Jean-Charles Faugère. A New Efficient Algorithm for Computing Gröbner Bases without Reduction to Zero (F5). *In T. Mora, editor, Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation ISSAC*, pages 75–83, 2002.

[FGLM93] Jean-Charles Faugère, Patrizia M. Gianni, Daniel Lazard, and Teo Mora. Efficient Computation of Zero-Dimensional Gröbner Bases by Change of Ordering. *J. Symb. Comput.*, 16(4):329–344, 1993.

[FMR09] Jean-Charles Faugère, Raphaël Marinier, and Guénaël Renault. Implicit Factoring with Shared Most Significant Bits. *preprint, personal communication*, 2009.

[FP83] Ulrich Fincke and Michael Pohst. A Procedure for Determining Algebraic Integers of Given Norm. In van Hulzen [vH83], pages 194–202.

[FR95] Matthew K. Franklin and Michael K. Reiter. A Linear Protocol Failure for RSA with Exponent Three. In *Presented at CRYPTO Rump Session*, 1995.

[Hås88] Johan Håstad. Solving Simultaneous Modular Equations of Low Degree. *SIAM J. Comput.*, 17(2):336–341, 1988.

[Hel85] Bettina Helfrich. Algorithms to Construct Minkowski Reduced and Hermite Reduced Lattice Bases. *Theor. Comput. Sci.*, 41:125–139, 1985.

[HM08] Mathias Herrmann and Alexander May. Solving Linear Equations Modulo Divisors: On Factoring Given Any Bits. In Josef Pieprzyk, editor, *ASIACRYPT*, volume 5350 of *Lecture Notes in Computer Science*, pages 406–424. Springer, 2008.

[HM09] Mathias Herrmann and Alexander May. Attacking Power Generators Using Unravelled Linearization: When Do We Output Too Much? In Mitsuru Matsui, editor, *ASIACRYPT*, volume 5912 of *Lecture Notes in Computer Science*, pages 487–504. Springer, 2009.

[HS07] Guillaume Hanrot and Damien Stehlé. Improved Analysis of Kannan's Shortest Lattice Vector Algorithm. In Menezes [Men07], pages 170–186.

[JM06] Ellen Jochemsz and Alexander May. A Strategy for Finding Roots of Multivariate Polynomials with New Applications in Attacking RSA Variants. In Lai and Chen [LC06], pages 267–282.

[JM07] Ellen Jochemsz and Alexander May. A Polynomial Time Attack on RSA with Private CRT-Exponents Smaller Than $N^{0.073}$. In Menezes [Men07], pages 395–411.

[JNT07] Antoine Joux, David Naccache, and Emmanuel Thomé. When e-th Roots Become Easier Than Factoring. In *ASIACRYPT*, pages 13–28, 2007.

[Jut98] Charanjit S. Jutla. On Finding Small Solutions of Modular Multivariate Polynomial Equations. In Nyberg [Nyb98], pages 158–170.

[Kan83] Ravi Kannan. Improved Algorithms for Integer Programming and Related Lattice Problems. In *STOC* [DBL83], pages 193–206.

[Kan87] Ravi Kannan. Minkowski's Convex Body Theorem and Integer Programming. In *Math. Oper. Res.* [DBL83], pages 415–440.

[Kat01] Stefan Katzenbeisser. *Recent Advances in RSA Cryptography.* Advances in Information Security 3. Springer US, 2001.

[Laz83] Daniel Lazard. Gröbner-Bases, Gaussian Elimination and Resolution of Systems of Algebraic Equations. In van Hulzen [vH83], pages 146–156.

[LC06] Xuejia Lai and Kefei Chen, editors. *Advances in Cryptology - ASIACRYPT 2006, 12th International Conference on the Theory and Application of Cryptology and Information Security, Shanghai, China, December 3-7, 2006, Proceedings*, volume 4284 of *Lecture Notes in Computer Science*. Springer, 2006.

[Len87] Hendrik W. Lenstra. Factoring Integers with Elliptic Curves. *The Annals of Mathematics*, 126(3):649–673, 1987.

[LHWL93] Arjen K. Lenstra and Jr. Hendrik W. Lenstra, editors. *The Development of the Number Field Sieve*, volume 1554 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 1993.

[LLL82] Arjen K. Lenstra, Hendrik W. Lenstra Jr., and László Lovász. Factoring Polynomials with Rational Coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.

[LR06] Gregor Leander and Andy Rupp. On the Equivalence of RSA and Factoring Regarding Generic Ring Algorithms. In Lai and Chen [LC06], pages 241–251.

[Lüb02] Frank Lübeck. On the Computation of Elementary Divisors of Integer Matrices. *J. Symb. Comput.*, 33(1):57–65, 2002.

[Mau92] Ueli M. Maurer. Factoring with an Oracle. In Rainer A. Rueppel, editor, *EUROCRYPT*, volume 658 of *Lecture Notes in Computer Science*, pages 429–436. Springer, 1992.

[Mau96] Ueli M. Maurer, editor. *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, volume 1070 of *Lecture Notes in Computer Science*. Springer, 1996.

[May06] Alexander May. *Skript zur Vorlesung Public Key Kryptanalyse.* Wintersemester 2005/06.

[McN07] John M. McNamee. *Numerical Methods for Roots of Polynomials, Part 1*, volume 14. Elsevier: Studies in Computational Mathematics, 2007.

[Men07] Alfred Menezes, editor. *Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings*, volume 4622 of *Lecture Notes in Computer Science*. Springer, 2007.

[Mey00] Carl D. Meyer. *Matrix Analysis and Applied Linear Algebra.* Cambridge University Press, 2000.

[MG02] Daniele Micciancio and Shafi Goldwasser. *Complexity of Lattice Problems: a Cryptographic Perspective*, volume 671 of *The Kluwer International Series in Engineering and Computer Science.* Kluwer Academic Publishers, Boston, Massachusetts, March 2002.

[Min96] Hermann Minkowski. *Geometrie der Zahlen.* Teubner-Verlag, 1896.

[MM82] Ernst W. Mayr and Albert R. Meyer. The Complexity of the Word Problems for Commutative Semigroups and Polynomial Ideals. *Advances in Mathematics*, 46(3):305–329, 1982.

[MR08] Alexander May and Maike Ritzenhofen. Solving Systems of Modular Equations in One Variable: How Many RSA-Encrypted Messages Does Eve Need to Know? In Ronald Cramer, editor, *Public Key Cryptography*, volume 4939 of *Lecture Notes in Computer Science*, pages 37–46. Springer, 2008.

[MR09] Alexander May and Maike Ritzenhofen. Implicit Factoring: On Polynomial Time Factoring Given Only an Implicit Hint. In Stanislaw Jarecki and Gene Tsudik, editors, *Public Key Cryptography*, volume 5443 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2009.

[NS05] Phong Q. Nguyen and Damien Stehlé. Floating-Point LLL Revisited. In Cramer [Cra05], pages 215–233.

[Nyb98] Kaisa Nyberg, editor. *Advances in Cryptology - EUROCRYPT '98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May 31 - June 4, 1998, Proceeding*, volume 1403 of *Lecture Notes in Computer Science*. Springer, 1998.

[Pau07] Franz Pauer. Gröbner Bases with Coefficients in Rings. *J. Symb. Comput.*, 42(11-12):1003–1011, 2007.

[Pom84] Carl Pomerance. The Quadratic Sieve Factoring Algorithm. In Thomas Beth, Norbert Cot, and Ingemar Ingemarsson, editors, *EUROCRYPT*, volume 209 of *Lecture Notes in Computer Science*, pages 169–182. Springer, 1984.

[RS85] Ronald L. Rivest and Adi Shamir. Efficient Factoring Based on Partial Information. In Franz Pichler, editor, *EUROCRYPT*, volume 219 of *Lecture Notes in Computer Science*, pages 31–34. Springer, 1985.

[RSA] RSA Laboratories. PKCS #1 v2.1: RSA Cryptography Standard.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[SB02] Josef Stoer and Roland Bulirsch. *Introduction to Numerical Analysis*. Springer-Verlag, New York and Berlin, 2002.

[Sho94] Peter W. Shor. Polynominal Time Algorithms for Discrete Logarithms and Factoring on a Quantum Computer. In Leonard M. Adleman and Ming-Deh A. Huang, editors, *ANTS*, volume 877 of *Lecture Notes in Computer Science*, page 289. Springer, 1994.

[Sho05] Victor Shoup. A Computational Introduction to Number Theory and Algebra. Cambridge University Press, 2005.

[Sil01] Joseph H. Silverman, editor. *Cryptography and Lattices, International Conference, CaLC 2001, Providence, RI, USA, March 29-30, 2001, Revised Papers*, volume 2146 of *Lecture Notes in Computer Science*. Springer, 2001.

[Sim83] Gustave J. Simmons. A "Weak" Privacy Protocol Using the RSA Crypto Algorithm. *Cryptologia*, 7(2):180–182, 1983.

[SM09] Santanu Sarkar and Subhamoy Maitra. Further Results on Implicit Factoring in Polynomial Time. *Advances in Mathematics of Communication*, 3(2):205–217, 2009.

[SW99] Uwe Storch and Hartmut Wiebe. *Lehrbuch der Mathematik, Band II: Lineare Algebra* . Spektrum Akademischer Verlag, 1999.

[vH83] J. A. van Hulzen, editor. *Computer Algebra, EUROCAL '83, European Computer Algebra Conference, London, England, March 28-30, 1983, Proceedings*, volume 162 of *Lecture Notes in Computer Science*. Springer, 1983.

[vH01] Mark van Hoeij. Factoring Polynomials and 0-1 Vectors. In Silverman [Sil01], pages 45–50.

[VZ95] Scott A. Vanstone and Robert J. Zuccherato. Short RSA Keys and Their Generation. *J. Cryptology*, 8(2):101–114, 1995.

[Wie90] Michael J. Wiener. Cryptanalysis of Short RSA Secret Exponents. *IEEE Transactions on Information Theory*, 36(3):553–558, 1990.

Acknowledgments

First of all, I would like to thank my supervisor Prof. Dr. Alexander May for giving me the opportunity to write this thesis, for inviting me to accompany him when he took up a new position in Bochum and, in particular, for the many and inspiring scientific discussions and the constant support at all stages of the thesis. Furthermore, I would like to say thank you to Prof. Dr. Hans Ulrich Simon for agreeing to act as co-referee. Moreover, I am grateful to Dr. Olivier Brunat for suggesting I should have a look at elementary divisors.
I have greatly enjoyed my time working at the Technical University of Darmstadt and at the Ruhr-University of Bochum. Thanks to the group of CDC in Darmstadt as well as the group of CITS and the other chairs of the HGI in Bochum who contributed to this. In particular, I would like to thank Christina Lindenberg and Roberto Samarone dos Santos Araújo for making me feel so welcome and for a great time sharing the office in Darmstadt. Thank you to Mathias Herrmann for more than two years of shared office time in Bochum, for discussing many scientific ideas or problems, but also for other discussions. Further, thank you, Mathias, for the thorough proofreading of my thesis, the helpful advice given on many occasions and for somehow managing to find enjoyment in tackling those little problems which to me just seemed annoying. Furthermore, I would like to thank Alexander Meurer and Enrico Thomae for proofreading a part of this thesis and for their helpful remarks. A special thanks to Christian Stinner for a very detailed proofreading although he had many other duties in the same period of time. Thank you, Christian, for your many helpful comments and for your encouragement.
Moreover, I would like to say thank you to my family and friends who have encouraged, supported and distracted me over the past years. Thank you to my parents and my brother for their constant support at all times, for their love and understanding.
Thank you to all the friends who have been there at one time or another and who contributed to so many enjoyable moments. In particular, I would like to thank Christiane for a longlasting friendship which has overcome all the changes in our lives, and for the opportunity to share our problems and frustrations as they arose. Although we were never really able to understand what the exact scientific problem of the other was, this helped me a lot. Thank you, Sarah, for all our intense discussions on nearly any topic (except for mathematics, maybe), for always being there in spite of any physical distance. Thank you, Tina, for so many true comments at exactly the right moments, for always joining in when it was once again time ”to drink some more tea” and for all those comforting laps of ”jogging”.
Thank you to all the people who made the past few years the way they were!